2

**Naval Research Laboratory**

Washington, DC 20375-5320

NRL/MR/1003--93-7396

# Neural Networks: An NRL Perspective

CHIP BACHMANN, PAUL BEY, JEREMY BROUGHTON, VICTOR CHEN, SHELDON GARDNER, BEHROOZ KAMGAR-PARSI, BEHZAD KAMGAR-PARSI, MOON KIM, FRANCIS KUB, KEITH MOON, ABRAHAM SCHULTZ, JOHN SCIORTINO, DEAN SCRIBNER, ANDREW SKINNER, WILLIAM TOLLES, JEFF WILLEY, AND SHELDON WOLK

*Working Group on Neural Networks*

September 3, 1993

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave Blank)* | 2. REPORT DATE<br>September 3, 1993 | 3. REPORT TYPE AND DATES COVERED |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>Neural Networks: An NRL Perspective | 5. FUNDING NUMBERS |
| 6. AUTHOR(S)<br>Chip Bachmann, Paul Bey, Jeremy Broughton, Victor Chen, Sheldon Gardner, Behrooz Kamgar-Parsi, Behzad Kamgar-Parsi, Moon Kim, Francis Kub, Keith Moon, Abraham Schultz, John Sciortino, Dean Scribner, Andrew Skinner, William Tolles, Jeff Willey, and Sheldon Wolk | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Research Laboratory<br>Washington, DC 20375-5320 | 8. PERFORMING ORGANIZATION REPORT NUMBER<br>NRL/MR/1003–93-7396 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
| 11. SUPPLEMENTARY NOTES | |
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution unlimited. | 12b. DISTRIBUTION CODE |

13. ABSTRACT *(Maximum 200 words)*

The subject of neural networks is introduced as a brief review and to provide perspective to the subject. A number of programs at the Naval Research Laboratory (NRL) are outlined to describe the general approach used and the anticipated benefits of neural networks. Observations and recommendations concerning the subject are made on the subject. This document serves as a base of information for additional internal discussions concerning opportunities for neural networks within NRL.

| | | |
|---|---|---|
| 14. SUBJECT TERMS | | 15. NUMBER OF PAGES<br>73 |
| Neural networks<br>Parallel processing<br>Parallel computers | Signal processing<br>Pattern recognition<br>Optimization methods | 16. PRICE CODE |

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

# CONTENTS

# PREFACE

In October 1992, a group of interested NRL researchers formed a working group to examine the questions and issues associated with the field of neural networks, examining both fundamental and applications aspects of this popular field. A number of presentations and round table discussions were held. Some external speakers were invited to augment the information examined. This exercise was both educational and broadening with respect to the utility and applicability of neural networks. An improved picture with respect to future aspects of neural networks was obtained and shared by the group. This document gives a number of the approaches examined by this working group during the course of these discussions.

# NOTATION

| | |
|---|---|
| $a_i$ | the i-th value of the input into a network; one subscript indicates that no classification is evident for the value - this element belongs to a vector |
| $a_{j\mu}$ | a value for "input" data corresponding to the j-th component of pattern $\mu$ |
| $A$ | a variable coefficient used in several different types of problems |
| $\mathbf{A}$ | the matrix of elements $a_{j\mu}$ |
| $\mathbf{A}^t$ | transpose of the matrix $\mathbf{A}$ |
| $\Delta\mathbf{A}$ | change in the parameters $a_i$ in a least squares iteration |
| $b_{j\mu}$ | the $j,\mu$-th component resulting from the operation of the weight matrix on the input data |
| $b^{(n)}{}_{i\mu}$ | the value of $b_{i\mu}$ as it appears in the n-th layer of a multi-layer network |
| $B$ | a variable coefficient used in several different types of problems |
| $C$ | a variable coefficient used in several different types of problems |
| $d_{ij}$ | derivative of a function at location i with respect to parameter j |
| $D$ | the value of a polynomial function in two variables defined in the text |
| $\mathbf{D}$ | the matrix of first derivatives |
| $\mathbf{D}^t$ | the transpose of matrix $\mathbf{D}$ |
| $\Delta\mathbf{Z}$ | matrix containing the difference between calculated and observed values, $z_i^o - z_i^c$ |
| $E$ | the sum of the squares of the differences between measured (or desired) and calculated values of the output of a network; using the LMS approach, $E = -(1/2)\Sigma_{\mu,i}(s_{i\mu} - \zeta_{i\mu})^2$ |

## NOTATION

| | |
|---|---|
| $\mathfrak{E}[\ ]$ | the average over a suitable time interval of the enclosed quantity [ ] |
| f() | is the nonlinear function transforming the product of the linear transformations; this is often called the "activation function" or "squashing function." The function which is frequently used is $$f(x) = 1/(1+e^{-\alpha x})$$ where $\alpha$ is a variable. The "hard limit" of this function is a step function, which is used for information of a "yes-no" nature. It may also be a "radial basis function" which selects a specific range of values. |
| $f_{ij}$ | the weighted value of the corrected output voltages for pixel located at position i-j in a focal plane array; $f_{ij} = (y_{i+1,j} + y_{i-1,j} + y_{i,j+1} + y_{i,j-1})/4$ |
| $f$ | the weighted value of the corrected output voltages for an unspecified pixel in a focal plane array (the subscripts i,j have been omitted for simplicity) |
| $f(a_1,a_2,\ldots a_n,x_{1i},x_{2i},\ldots x_{mi})$ | a function of n parameters and m independent variables corresponding to the i-th measure of this function |
| $F_i$ | designation of the i-th layer in a network |
| $g(x\vert k)$ | a conditional density function chosen to represent the statistical values of a sequence in the variable x, for class k |
| $G_{ij}$ | the gain of a pixel at location i-j in a focal plane array |
| $G_n$ | value of the n-th iteration for the gain in a focal plane array (the subscripts i,j are omitted for simplicity of notation) |
| G(x,y) | Gabor function of variables x and y |
| **H** | the matrix of second derivatives of a function: the Hessian matrix |
| $\mathcal{H}_j(x)$ | the j-th Hermite polynomial in variable x |
| K | the number of clusters appearing in the Traveling Salesman Problem |

## NOTATION

| | |
|---|---|
| $kT$ | thermal energy: Boltzmann constant times the temperature |
| $N$ | the number of patterns appearing in the Traveling Salesman Problem |
| $O_{ij}$ | an offset correction applied to the voltage of pixel located at position i-j in a focal plane array |
| $P_i()$ | probability for the occurrence of the contents inside () for the i-th example in question |
| $p_z(z\|i)$ | the probability density function for variable z belonging to class i |
| $r_i$ | center of mass of an image having a vector of intensities at row (or column) i, defined as $r_i = \Sigma_j j\rho_{ij}$ where $\rho_{ij}$ is the intensity of element j in the row (or column), suitably normalized |
| $S$ | the value of a polynomial function in one variable as defined in the text |
| $s^{(n)}_{i\mu}$ | the value of $s_{i\mu}$ as it appears in the n-th layer of a multi-layer network |
| $s_i^{(n)}$ | the i-th neural value in layer n; it does not correspond to any particular class, hence only one subscript is required |
| $s_{i\mu}$ | the value of a "neuron" in a given layer after the activation function has been applied to the output of a previous layer; it is equivalent to $f(b_{i\mu})$ |
| $T$ | the value of a polynomial function in three variables defined in the text |
| $V_i$ | voltage at a particular position i in a circuit |
| $V_{pi}$ | (1) a value of 1 or 0 in the Traveling Salesman Problem, representing a route between locations p and i if it is 1; (2) a voltage between points i and p between two positions on a bridge (for cellular network example) |
| $w_{ij}$ | a weight connecting the j-th input element to the i-th node (or "neuron") in the next layer of a network |
| $w^{(n)}_{ij}$ | the value of $w_{ij}$ as it appears in the n-th layer of a multi-layer network |

# NOTATION

| | |
|---|---|
| $\Delta w_{mk}$ | a change in the weight for element m in class k |
| $\mathbf{W}$ | the matrix of elements $w_{ij}$ |
| $\mathbf{W}_k$ | the value of the matrix $\mathbf{W}$ after the k-th iteration |
| x | a variable used for a number of purposes, defined in the immediate area under discussion |
| $x_{ij}(\Phi_{ij})$ | the voltage from pixel i-j in a focal plane array which is exposed to a flux $\Phi_{ij}$ falling on that location |
| y | the value of a variable or function, described in the immediate text |
| y | corrected output voltage from an unspecified pixel in a focal plane array |
| $y_{ij}$ | corrected output voltage from pixel at location i-j of a focal plane array |
| $z_i^c$ | calculated value of a function corresponding to the i-th observed value of that function |
| $z_i^o$ | i-th observed value of a set of data |

## GREEK SYMBOLS

| | |
|---|---|
| $\alpha$ | an adjustable parameter, used in a number of algorithmic approaches to adjust the rate of convergence of network values |
| $\delta_{ik}$ | the Kronecker delta, = 1 if i = k, and = 0 otherwise |
| $\Delta_{k\mu}$ | a mathematical relationship simplifying the notation for the relationship involving back propagation; $\Delta_{k\mu} = (s_{k\mu} - \zeta_{k\mu})\, f'(b_{k\mu})$ |
| $\gamma$ | a variable coefficient, typically empirically determined, used for various purposes (such as adjust rate of convergence of a given procedure) as defined in the immediate text |
| $\eta$ | an adjustable parameter altering the rate of convergence, used for a variety of networks and convergence approaches |
| $\phi$ | the angle between two vectors in n-dimensional space |

# NOTATION

| | |
|---|---|
| $\theta$ | used for an angle, specified in immediate text |
| $\theta_i$ | the "offset," a constant value added to the output of layer i before applying the "activation function" |
| $\omega$ | angular frequency |
| $\Omega_x$ | symbol for an n-dimensional space for variables x |
| $\zeta$ | the matrix of elements $\zeta_{i\mu}$ |
| $\zeta_{i\mu}$ | experimental data (or data to be emulated by a network) for point i belonging to class $\mu$; alternatively, it is a "desired" value which a network is asked to "fit." |

# NEURAL NETWORKS: AN NRL PERSPECTIVE

## 1. INTRODUCTION

Heard from frustrated practitioners:
*Neural networks are the second-best way to solve any problem*[1].

Computation and/or information processing with neural networks has been of interest for many years in association with the mechanisms involving memory and adaptation in biological systems. Examination of biological systems can provide insight and inspiration for new approaches to solving mathematical problems. The field of neural networks has been inspired through this approach, along with observations about mathematical properties which have proven sufficiently intriguing to attract considerable attention. Initial research began in the 1940's in which computers inspired by brain-like architectures were discussed. Such researchers included, among others, John von Neumann, who wrote several books on the subject. Brief histories of neurocomputing involving many notable accomplishments by early pioneers are available in a number of texts[2].

The basic concept behind neural networks involves processing at a local center (or node) and interconnections to other local centers through which information is shared and further processing can take place. The perceptron was perhaps the first "neurocomputer" conceived and built in the late 1950's (Frank Rosenblatt, 1957); it is based on linearly separable criteria for classification (this is discussed in the text). An electromechanical system was actually built to recognize characters using these principles. Restrictions to problems involving linear separability limited the utility and applicability of this construct. By the mid-1960's interest in neurocomputing began to wane.

1 There are, in fact, a number of examples where the use of neural networks have been shown to be superior to the conventional techniques which have been applied to problems.

2 See, for example, *Neurocomputing*, R. Hecht-Nielsen, Addison-Wesley Publishing Co., NY (1990). NRL Library QA 76.5 .H4442 (1990).

Manuscript approved July 30, 1993.

## INTRODUCTION

Two notable events revived interest in the field. First, Hopfield[3] published an article in which he described a relatively simple mathematical algorithm in which the input/output relationship associated with a mathematical data set appeared to iterate towards a local minimum; this could be used to "recognize" the pattern associated with the input data. Second, the algorithm for back propagation[4] set the stage for optimization in a wide variety of networks. The concept of "learning" by networks became a topic of considerable interest. This development occurred at a time of rapid increase in computational capabilities with decreased cost, and when conventional approaches for computation-intensive problems such as machine learning were approaching roadblocks. The ease of designing new algorithms and executing them on increasingly powerful personal computers has attracted a large number of practitioners. A great number of investigators introduced variations on the theme, and an explosion of technical publications followed.

A number of research programs existing at the time when neural networks became popular found it beneficial to "relabel" ongoing work with titles involving "neural network." With increased sponsor interest in the field, the attraction of these existing programs accelerated still further the appearance of rapid growth in the field. International Conferences on Neural Networks attracted hundreds of papers annually. Great interest peaked in the late 1980's, and there is currently a more thoughtful process of sifting and assessing the impact and future opportunities offered by the techniques introduced by this field.

The algorithms which have developed are useful in problems involving signal and image processing as well as detection, classification, and combinatorial problems. These techniques appear to have been introduced into a large number of applications. In a number of cases they appear to offer the most convenient formulation for specific problems. In other cases, even though :hey have been shown to give reasonable solutions to many problems;

3 Hopfield, J., *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences 79, 2554-2558 (1982).

4 D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*, in D. E. Rumelhart and J. L. McClelland (eds), Parallel Distributed Processing, Volume 1, MIT Press (1986).

alternative approaches to some problem have proven superior. A large number of studies have pointed out the performance of networks for specific applications without comparing them with competitive algorithms. The assessment of the true advantages of these methods continues and is a subject of considerable interest.

Even through the performance of neural networks may not surpass that of standard linear or nonlinear signal processing methods, neural networks offer some potentially valuable advantages that make them important in many application areas. Two potential advantages include:

a) Reduced time and effort in deriving complex algorithms and writing code. In some cases standard approaches do not even exist or would require special expertise or break-through concepts. Readily available neural network algorithms properly applied offer an immediate and cost-effective alternative.

b) Reduced digital processor size from cubic feet to wafer-scale or micro-chip sizes operating at very low power, and at the same time increasing the processing power by several orders of magnitude. In essence, new digital processors on a single wafer could perform with processing power greater than that of a larger system such as a CRAY or the Connection Machine.

A number of disadvantages should be kept in mind regarding neural networks. These include basic problems such as convergence and training efficiency. In addition, simply because a problem may be approached by using a neural network does not mean that the neural network is the best approach to implement. Some experience with the efficiency of problems may help in the choice of algorithmic approach. Generally, except for problems in which considerable complexity is involved, if the fundamental relationships for a problem can be expressed, it is often better to use these fundamental relationships rather than resort to a neural network.

# 2. BACKGROUND

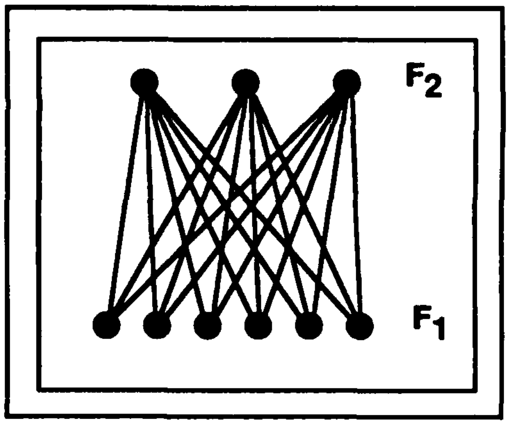
The fundamental behavior of neural networks was a subject of interest for a number of years before the excitement of the early 1980's[5]. The linear perceptron, for example, provides a useful linear transformation of a vector array comparable to a simple matrix multiplication. The fundamental relationship between "input" and "output" elements of a linear perceptron is given by[6]

$$\zeta_{i\mu} = \Sigma_j w_{ij} a_{j\mu} \qquad (2\text{-}1)$$

where $a_{j\mu}$ and $\zeta_{i\mu}$ are the j-th input and i-th desired output "voltages" (or values) of a network representing pattern $\mu$. The problem as frequently stated is to determine the elements $w_{ij}$ to satisfy this equation. This represents simple matrix multiplication:

$$\zeta = \mathbf{WA} \qquad (2\text{-}2)$$

where the upper case bold symbols without subscripts indicate matrices. The input and output matrices **A** and $\zeta$ are known, and the matrix **W** is to be determined. In the custom of the literature on neural networks, an illustration of this operation is given in the accompanying figure. The input stage is labelled as $F_1$, and the output stage is labelled $F_2$. Information thus "travels upward" as the mathematical operations take place. The matrix **A** is generally not square; hence, the inverse of **A** does not generally exist. However, an apparently straightforward way to solve this is by forming the pseudoinverse of the matrix **A**:



5 J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, Redwood City, CA (1991). NRL Library QA 76.5 .H475 1991.

6 Y. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley Publishing Company, Inc., New York, NY (1989). NRL Library TK 7882 .P3

$$W = \zeta\ A^t\ (AA^t)^{-1} \tag{2-3}$$

where $A^t$ is the transpose of $A$. In some cases the pseudoinverse may not exist, whereupon this process will not give an answer. In general, since $A$ does not have an inverse, the value of $WA$ as represented by equation (2-3) is $\zeta A^t(AA^t)^{-1}A \neq \zeta$, and the solution obtained by this method does not lead to an equality. Further, real systems such as biological systems manage to find extrema and determine relationships between stimulus and response without inverting matrices, but rather by adjusting certain parameters to find such extrema. This inherent "feel" for finding correct weighting interrelationships has been one factor in stimulating the current interest in neural networks.

Attempts to converge on values of $W$ iteratively may be illustrated with one approach known as the "Widrow-Hoff procedure" (sometimes referred to as "Hebbian learning") in which successive values of $W$ (the k-th iteration is indicated as $W_k$) are obtained by updating the matrix of weights $W_{k+1} = W_k + \Delta W$

where
$$\Delta W = \alpha(\zeta - W_k A)A \tag{2-4}$$

and $\alpha$ is a small number chosen to allow convergence. This procedure in effect adds an increasing amount of the "training vectors" in the columns of $A$ with increased weight to those rows of $W_k$ which most closely resemble the column vectors in $A$. In reality, for many cases of interest, the iterations do not converge. Many alternatives have been reported for such learning methods. Alternatives include constantly renormalizing the vectors in $W$. Depending on the type of network, the "learning rules" vary considerably[7].

Some attempts to improve the linear perceptron introduce "multiple layers" in which the matrix $W$ is represented as a product of two matrices. However, since the product of two matrices is simply another matrix, nothing new was added to the perceptron as long as linear transformations were involved.

[7] J. M. Surada, *Introduction to Artificial Neural Systems*, West Publishing Co., NY (1992). ISBN 0-314-93391-3

## BACKGROUND

Adjustment of the "weights" in the transformation matrix **W** lead to relationships between known values of **A** and $\zeta$. However, it was pointed out that the linear perceptron had serious fundamental limitations. It could not even solve (i.e. find a suitable matrix **W**) the XOR problem, in which the logical input/output relationship is:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \zeta = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}.$$

Attempting to obtain a matrix **W** which agrees with **A** and $\zeta$ is equivalent to finding a linear discriminant (a straight line separating two classes) in two-dimensional space. The two classes "1" and "0" given by these 4 points in two-dimensions: $(1,1)\rightarrow 0$, $(1,0)\rightarrow 1$, $(0,1)\rightarrow 1$, and $(0,0)\rightarrow 0$. They are not separated by a plane, hence there is no linear discriminant for solving this problem. Interest in the linear perceptron faded until subsequent nonlinear transformations and the use of multiple layers were shown to overcome this problem. This is one of the reasons why the current neural network models are of interest today.

## 2.1. NEURAL NETWORKS: BEYOND THE PERCEPTRON

A great many variations of neural networks have been introduced in the literature and examined[8]. Several networks illustrating a number of fundamental principals will be discussed here.

### 2.1.1. Single Layer Networks

A fundamental difference among the many attempts to use the linear perceptron and the current interest in neural networks may be illustrated by simply introducing a nonlinear transformation between successive operations in the perceptron model. This nonlinear transformation, as it turns out, provides a great deal of flexibility in developing the relationships between A and $\zeta$. With this alternative approach, the XOR problem is readily solved. In reality, the use of nonlinear transformations combined with the use of "hidden layers" (to be discussed subsequently) provides a highly flexible function in n-

[8] See, for example, *DARPA Neural Network Study*, AFCEA International Press, Fairfax, VA, October 1987 - February 1988. NRL Library: UG 479 .D37 (1988).

dimensional space. One of the challenges which remains is to find efficient algorithms for convergence in order to obtain the desired relationships.

The single layered network may be described briefly. Let $a_{j\mu}$ be an initial set of patterns (j represents the j-th element of the $\mu$-th pattern), and let $w_{ij}$ be an initial choice of weights chosen at random or according to some other rationale which introduces some relationship between the "input" and "output." Calculate the actual relationship between "input" and "output" in the same manner as for a linear perceptron:

$$b_{i\mu} = \Sigma_j w_{ij} a_{j\mu}. \tag{2-5}$$

Introduce the "activation function" f(x) [which may be $1/(1+e^{-x})$ or some other "squashing function"] and a bias $\theta_i$ to modify the previous operation:

$$s_{i\mu} = f(b_{i\mu}) = f(\Sigma_j w_{ij} a_{j\mu} + \theta_i) \tag{2-6}$$

Assume $\zeta_{i\mu}$ is the desired value. Define the error or "cost function" E between the calculated and observed values:

$$E = (1/2)\Sigma_{\mu,i}(\zeta_{i\mu} - s_{i\mu})^2 \tag{2-7}$$

This value is then minimized with respect to the weights $w_{km}$, a process referred to as "learning." The learning procedure described below updates the weights by making use of the derivatives of E with respect to each weight $w_{km}$:

$$(\partial E/\partial w_{km}) = -\Sigma_{\mu,i}(\zeta_{i\mu} - s_{i\mu})(\partial s_{i\mu}/\partial w_{km}). \tag{2-8}$$

By equations (2-5) and (2-6)

$$(\partial s_{i\mu}/\partial w_{km}) = (\partial f/\partial b_{i\mu})(\partial b_{i\mu}/w_{km}) = f'(b_{i\mu})\, a_{m\mu}\delta_{ik} \tag{2-9}$$

where $\delta_{ik} = 1$ if $i = k$ and 0 otherwise. Consequently,

$$(\partial E/\partial w_{km}) = -\Sigma_\mu (\zeta_{k\mu} - s_{k\mu})\, f'(b_{k\mu})\, a_{m\mu} = \Sigma_\mu \Delta_{k\mu}\, a_{m\mu} \tag{2-10}$$

where

$$\Delta_{k\mu} = (\zeta_{k\mu} - s_{k\mu})\, f'(b_{k\mu}). \tag{2-11}$$

The algorithm frequently used to find a minimum in the value of E is that of "steepest descent" or "gradient descent," adjusting the weights $w_{km}$ such that

$$\Delta w_{km} = -\eta(\partial E/\partial w_{km}). \tag{2-12}$$

where $\eta$ is an adjustable parameter altering the rate of convergence for a given problem, and whose magnitude is determined by experience with a given type of problem. The virtue of the steepest descent method is that, for small incremen's $\eta$, it guarantees a lower value of E if the derivative is not at a minimum, since, by (2–12),

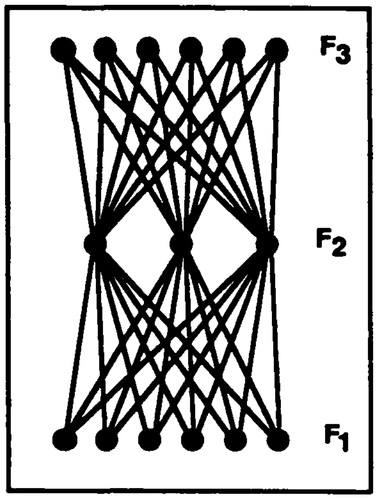
$$\Delta E = (\partial E/\partial w_{km})\Delta w_{km} = -\eta(\partial E/\partial w_{km})^2 < 0. \tag{2-13}$$

The disadvantage of the steepest descent method is that it is an inefficient procedure for rapid convergence. Consider an alternative, for example, of finding an extremum for a quadratic surface. The usual relationship which would converge in one iteration is $\Delta w = \Delta E/(\partial E/\partial w)$, which guarantees convergence in a single iteration. This is a superior method if the surface is quadratic, which is the case for any surface close to an extremum (expansion by Taylor's series). A number of alternative methods are becoming popular for determining extrema (see Appendix III).

The procedure described in (2-12) may be readily generalized to handle multiple layers. The convergence to a minimum using this procedure may be excruciatingly slow for some problems, depending on the spread of eigenvalues for the matrix $\Delta$. Alternative approaches to obtain minima have been the subject of numerous papers.

### 2.1.2. Multiple Layer Networks

The steepest descent method provides convergence and improved solutions relative to the linear perceptron for a number of problems, including the XOR problem. Additional functional flexibility and/or ease of convergence is introduced with multiple layers. The layer referenced in each equation will be introduced as a superscript in parentheses, with $b_{i\mu} \equiv b^{(1)}{}_{i\mu}$; thus, for the presence of a second layer,



$$b^{(2)}{}_{i\mu} = \Sigma_j w^{(2)}{}_{ij}\, s^{(1)}{}_{j\mu} \qquad (2\text{-}14)$$

and

$$s^{(2)}{}_{i\mu} = f(b^{(2)}{}_{i\mu}). \qquad (2\text{-}15)$$

The "energy" may be evaluated in a manner similar to that outlined by equations (2-7) through (2-11):

$$(\partial E/\partial w^{(2)}{}_{mn}) = -\, \Sigma_\mu\, (s^{(2)}{}_{m\mu} - \zeta_{m\mu})\; f'(b^{(2)}{}_{m\mu})\, s^{(1)}{}_{n\mu} = \Sigma_\mu\, \Delta^{(2)}{}_{m\mu}\, s^{(1)}{}_{n\mu}. \qquad (2\text{-}16)$$

The values of $w^{(2)}{}_{mn}$ may be adjusted using the steepest descent method. However, the adjustment of $w^{(1)}{}_{mn}$ must use a relationship

$$(\partial E/\partial w^{(1)}{}_{pq}) = -\, \Sigma_{i,\mu}\, (s^{(2)}{}_{i\mu} - \zeta_{i\mu})\; f'(b^{(2)}{}_{i\mu}) w^{(2)}{}_{ip} f'(b^{(1)}{}_{p\mu}) a_{q\mu}$$

$$= -\, \Sigma_\mu\, \{ f'(b^{(1)}{}_{p\mu})[\Sigma_i\, (s^{(2)}{}_{i\mu} - \zeta_{i\mu}) f'(b^{(2)}{}_{i\mu}) w^{(2)}{}_{ip}] \}\, a_{q\mu}$$

$$= -\, \Sigma_\mu\, \Delta^{(1)}{}_{p\mu}\, a_{q\mu} \qquad (2\text{-}16)$$

where the definition of $\Delta^{(1)}{}_{p\mu}$ may be taken from the equalities in (2-16). This technique of iteratively generating successive values of the weights is known as "back propagation." A layer which does not have input or output "nodes" is referred to as a "hidden layer," possible only in networks having three or more layers.

## BACKGROUND

Cybenko[9] showed that the linear combination of functions inherent in neural networks was equivalent to a single very flexible function, able to simulate any well-behaved functional form in n-dimensional space. In fact, it was shown that "any continuous function can be uniformly approximated by a continuous neural network having only one internal, hidden layer and with an arbitrary continuous sigmoidal nonlinearity."

As with single layered networks, the approach used to "learning" is equivalent to adjusting the weights of neural networks to find minima in E with a consequent reproduction of a desired input-output behavior. Once a network has been "trained" by such procedures, additional information is processed with the weights are kept constant; the classification or desired response from the network is then obtained.

The simplicity of the multiple layered concept and the great generality of problems to which it may be applied has attracted a large number of participants into the neural network community. The possibility of training a machine/algorithm to solve a wide variety of problems which are not well understood, without having to do programming, provides a great attraction. A number of impressive demonstrations of successful solutions have been obtained.

Beyond this straightforward introduction, networks have been introduced which are far more complex and which demonstrate an uncountable number of variations. Interconnections among "neurons" (the ${s^{(n)}}_i$ values) may be designed to pass over more than one layer (whereupon the common matrix notation fails to represent the mathematics adequately), and feedback connections (from ${s^{(n)}}_i$ to ${s^{(m)}}_j$, where $m<n$) may also be introduced. These additional complexities add many variations to the possible functional dependence associated with neural networks. Such combinations have been used to solve a great number of problems of practical interest. However, with this popularity came a lack of rigor in notation and approach, since the fundamentals were readily amenable to variations. Solutions to problems were examined and reported in a great variety of fields. A significant shortcoming of many of these studies is a lack of comparison with alternative approaches or a solid understanding of the merits or limits of the processes examined. If false

---

[9] G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function*, Math. Control Signals Systems 2, 303-314 (1989).

extrema are found through some convergence procedure, there is no guarantee that the solution thus obtained is the global (and best) minimum. The true performance improvement associated with neural networks is still being evaluated and compared with other more conventional methods.

### 2.1.3. Unsupervised Classification and Self-Organizing Maps

One useful application of neural networks involves the classification of items based on similar characteristics, where the pattern, number of classes, and membership within each class is determined through execution of an algorithm. Various approaches to this problem have been examined.

A popular approach found useful in problems encountered at NRL is the Learning Vector Quantizer (LVQ) introduced by Kohonen. In this approach, features corresponding to each member to be classified are mapped into n-dimensional feature space. A comparison is made between the length of each vector with initial vectors chosen at random (or with some pattern in mind prior to the classification task). A small quantity of each chosen vector is added to the initially random vector which it most closely resembles. This procedure is iterated for each member to be classified. The resulting vectors are found to follow directions characterizing each class of data. An illustration of this method is presented in the automatic ESM system program described in the applications section of this document.

Alternative classifiers such as the Linde-Buzo-Gray (LBG) classifier typically partition feature space with an initial set of assumed cluster centers, and determine the centroid of all points falling within the chosen partitions. Subsequent iterations first move the centroid of the assumed cluster centers to that existing within each partitioned space, then subsequently readjust the partition locations based on the new location of each centroid. Iterative steps to a self-consistent set of partitions and centroids serve to define each cluster.

## 2.2. TYPES OF NETWORKS

### FEEDFORWARD AND FEEDBACK NETWORKS

This type of network contains connections in which each layer of nodes is connected only to the preceding layers. Such an arrangement appears to be the most common in use today. Alternative topologies include feedback as well as feedforward connections. Feedback certainly introduces additional complexity; however, the back propagation model continues to give some degree of convergence on extrema when used with neural networks having feedforward and/or feedback.

## HOPFIELD NETWORK

The original highly cited paper by Hopfield[10] introduced concepts and algorithmic behavior which stimulated considerable subsequent activity. The formulation was clearly modelled after concepts associated with neural activity, and much of the notation and language followed these concepts. Hopfield was concerned with the recognition of patterns, and proposed a specific initial choice of weights and activation function which provided a surprisingly effective (although limited) pattern recognition capability. The basic algorithm for feedforward "activation" of each "neuron" is the same as that featuring equation (2-6). The activation function used is the hard limit, represented by the step function:

$$f(x) = \begin{cases} 0 \text{ if } x<0 \\ 1 \text{ if } x>0 \end{cases}$$

All neurons have the values of either $s_{ik} = 1$ or 0 which represent amplitudes for an initial pattern belonging to class k. The initial choice of weights is given as

$$w_{ij} = \Sigma_k \, (2s_{ik} -1)(2s_{jk}-1), \; i \neq j$$

with the additional restriction that the diagonal elements are zero:

$$w_{ii} = 0.$$

Iteration of the "neuron values" takes the form

$$s_{im}^{new} = f\{\Sigma_j w_{ij} s_{jm}\} = f\{\Sigma_j \, \Sigma_k \, (2s_{ik} -1)(2s_{jk}-1)s_{jm}\}$$

10 J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences **79**, 2554-2558 (1982).

$$= f\{\Sigma_k (2s_{ik} - 1)\Sigma_j (2s_{jk} - 1)s_{jm}\} \approx f\{\Sigma_k (2s_{ik} - 1)\delta_{km}N/2\} \approx s_{im}$$

where, for patterns which are not closely related, the sum over j represents a product of nearly orthogonal vectors with length N/2 (about half of the values of $s_{jk}$ will be zero). Thus, the calculation of the neuron values for a given pattern gives rise to a pattern most closely resembling the input to the network, an interesting result. Adjustment of the weights using the previously outlined approaches will generate a "learned" network able to "purify" an input pattern, giving an output more closely resembling the pattern to which the network has been programmed. The result of this process is to provide a decision tool able to recognize and manipulate an input pattern according to a limited set of preprogrammed patterns.

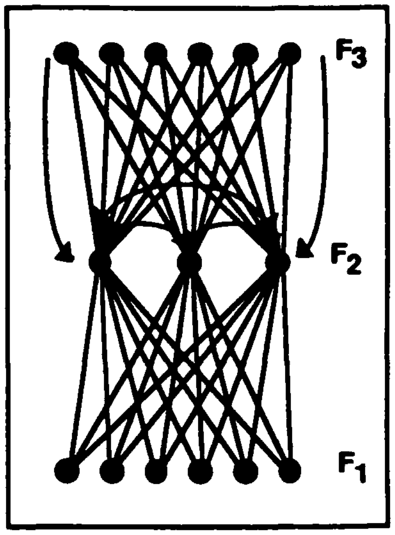
## KOHONEN SELF-ORGANIZING NETWORK

The Kohonen network is a self-organizing network used for classification [Footnote 6]. Each of a series of pre-existing events is "topologically mapped" with a set of characteristic parameters, forming an ordered array of data. A discriminant is introduced to assign a "merit" to each newly introduced event relative to each pre-existing event. An optimum pre-existing event is selected through this process as that with the greatest merit. After this selection process an adaptive process occurs which adjusts the parameters corresponding to the pre-existing value(s) by incorporating those of the newly selected event.

A frequently encountered application of the Kohonen network is to classify a sequence of observed vectors, each belonging to n classes characterized by their similarity (spatial closeness) in N-dimensional space. This is known as the Learning Vector Quantizer (LVQ). A random set of vectors may be initially chosen, where the number of random vectors is greater than n. The distance between each observed vector and each of the initial random vectors may be calculated in sequence, and the random vector most closely resembling (nearest distance) the observed vector is chosen. It is then modified by the addition of a fraction of the observed vector. This process is repeated for each observed vector in sequence. This produces n vectors which emerge from the initial random selection. The resulting n vectors exhibit average characteristics of the vectors belonging to each class. An example is demonstrated with the automatic ESM recognition section in this document.

## ART (ADAPTIVE RESONANCE THEORY)

Proposed by Grossberg[11], ART is capable of switching between the learning mode and the stable (classification) mode. This is illustrated by the accompanying figure[12]. Input at layer $F_1$ is fed into the next layer, $F_2$, which, in addition to the usual activation function, has "inhibition," in which the adjacent nodes interact with one another (illustrated by the curved arrows at the $F_2$ level). Inhibition may be defined by the relationship



$$s_i^{(n)} = s_i^{(n)} - \alpha \Sigma_{j \neq i} s_j^{(n)}$$

where $\alpha$ is an adjustable parameter. Such a relationship tends to increase the larger values of $s_i^{(n)}$ and decrease the smaller values. As such, inhibition serves to select the more "important" of the nodes in a manner which resembles the simple mathematical operation of finding the maximum. In addition, the output layer at $F_3$ is fed back to the hidden layer $F_2$ in order to stabilize the existing pattern which is developed during the "learning" process.

This arrangement has the advantage of accepting additional information from the input while stabilizing the overall pattern through feedback from the output layer. The degree of stabilization may be adjusted heuristically by changing the degree of feedback and inhibition.

11 S. Grossberg, *Nonlinear Neural Networks: Principles, Mechanisms, and Architectures*, Neural Networks, 1, 17-61 (1988).

12 G. A. Carpenter, *Neural Network Models for Pattern Recognition and Associative Memory*, Neural Networks 2, 243-257 (1989).

Such a network appears to incorporate a number of features reported in many articles in the literature. The construction of such networks may accomplish a goal given sufficient effort, but design and parameter adjustment appears to be highly dependent on the specific problem under investigation.

## 2.3. NEURAL NETWORKS AND STANDARD METHODS

Many manipulations performed by neural networks resemble algorithms and methods used for years in signal processing. The differences are clear, however, and should be recognized. Unfortunately, in many cases, practitioners involved with neural networks are not always aware of these previously developed techniques. One recent book recognizes the importance of establishing a framework for the design of networks[13]. A recent paper shows that the optimal adjustment of weights in a single-layer perceptron is equivalent to the Wiener solution for signal analysis[14]. The relationship between neural networks and partial least squares modeling has been recently investigated[15]. Progress may be improved if the fundamental building blocks allow future efforts to expand from a well-founded base of fundamentals.

The operations of neural networks resemble those of alternative methods to accomplish similar or related objectives. For example, the simple multiplication of a matrix and a column vector is equivalent to a recognition operation. If, for example, the rows of a matrix $w_{ij}$ and the components of a column vector $a_j$ are normalized, then the product is the inner product, $\Sigma_j w_{ij} a_j = \cos \phi$, the direction cosine between two vectors in n-dimensional space. The row of the product column vector which is most nearly equal to that of the input column vector is the index for the element having the greatest value. Subsequent application of the "squashing function" is another way of selecting (giving greater weight) the largest value relative to the other

13 T. Hrycej, *Modular Learning in Neural Networks*, John Wiley & Sons, Inc., NY (1992). NRL Library QA 76.87 .H78, ISBN 0-471-57154-7.

14 A. Feuer and R. Cristi, *On the Optimal Weight Vector of a Perceptron with Gaussian Data and Arbitrary Nonlinearity*, IEEE Transactions on Signal Processing, **41**, 2257-2259 (1993).

15 S. J. Qin and T. J. McAvoy, *Nonlinear PLS Modeling using Neural Networks*, Computers Chem. Engng, **16**, 379-391 (1992).

values present. Alternatively, "inhibition" is equivalent to increasing the largest value while diminishing smaller values of a selected set of numbers. Both of these nonlinear operations may be compared to the simple function of finding the maximum value of numbers in a column vector, setting it equal to unity, and setting the values of the other elements to zero. The operation of taking the inner product is also equivalent to forming a matched filter, a concept used in signal processing for years. The cross-correlation function is the optimal filter with Gaussian noise present, and, again, this is equivalent to the operations described above.

Thus, several standard signal processing techniques offer a way to find a vector most similar to another by evaluating the angle between them in n-dimensional space. Self-organizing methods by Kohonen likewise determine the similarity between vectors in n-dimensional space, only, in these self-organizing methods, the distance between two vectors is the measure of merit rather than the angle between the vectors.

Beyond the operation of classification outlined above, the vector analogy may extended to the operation of "purifying a pattern" recognized with a Hopfield network (and related networks). Such an operation may be perceived as an additional matrix operation following the *classification* operation. Once the most likely choice has been made, and a classification of the input vector has been determined, a subsequent multiplication by the appropriate matrix will generate a predefined pattern (or image) perfectly. An alternative relationship for generating these images is possible if the vectors for these images are nearly orthogonal, as pointed out in section 2.2.

Appendix II outlines the standard method of least squares and the use of the pseudoinverse as a means of performing that operation. The pseudoinverse has been recognized as having properties of a memory array[16] and is capable of giving directly the operations which recognize selected patterns stored in the pseudoinverse. The output array from such an operation consists of 1's and 0's based on the relationship between input and the stored information. Thus, classification decisions from these operations are clear and do not require nonlinear operations. More complex behavior

[16] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps*, Addison-Wesley Publishing Co., NY (1992). NRL Library QA 76.87 .R58; ISBN 0-201-55443-7.

may be simulated using nonlinear least squares techniques. This is generally done with iterative techniques involving linear approximations for each iteration. Iterative least squares techniques and the methods of multiple layer neural networks may indeed have some similarities; however, a careful analysis of the two techniques remains to be made.

## 2.4. HARDWARE IMPLEMENTATION

With the explosion of interest in neural networks is the introduction of specially designed chips and parallel processors. The ability to calculate numerical values at one processing unit using input from neighboring processing units on a single chip set the stage for efficient hardware implementation of network models. The combination of new algorithms along with the quantum leap in computational ability is one reason to expect advances which may prove quite powerful as advanced software and hardware combine in a synergistic manner.

One recent book discusses the various architectures available today for neural network calculations[17]. A number of new chips have been introduced which are patterned with the Single-Instruction Multiple Data (SIMD) architecture. From 1 to 512 input and arithmetic channels may be used to operate on the input vectors. Some architectures operate on two-dimensional arrays. Both fixed and floating point arithmetic are available from manufacturers today. It is important to note that, using this hardware, the number of weight updates per second for many neural network operations exceed the fastest single CPU performance in supercomputers today by 1 - 2 orders of magnitude. Thus, although several neural network algorithms seem computationally inefficient, hardware implementation may provide superior performance for some problems. Examples will be forthcoming and will be tested over the next few years.

[17] K. W. Przytula and V. K. Prasanna, *Parallel Digital Implementation of Neural Networks*, Prentice Hall, NJ (1993). NRL Library QA 76.87 .37; ISBN 0-13-649161-8.

# 3. APPLICATIONS OF NEURAL NETWORKS

A large number of applications using neural networks have been examined. Good references include the DARPA study of 1988 [Footnote 8] and the overview by Simpson[18]. The list is heavily weighted towards signal processing, pattern recognition, classification, and learning techniques for motion optimization (such as robotic action).

Several classification problems may be readily attacked using self-organization network algorithms. This is particularly true when the features to be recognized are represented by a vectors in n-dimensional space (such as the Fourier components of a time-dependent signal). The original classification by Kohonen was to produce a Finnish phonetic typewriter.

Signal processing, speech recognition, and similar challenges having data in the time or spatial dimension may be pursued with a multilayer perceptron with back propagation or some similar approach. Recognition of signals involving sonar, radar, speech, etc. has been shown to be successfully attacked using these approaches. Application of signal recognition techniques to problems such as helicopter gear box fault determination has been successful. Pattern completion is a form of signal processing in which a network has the ability to shape an input signal to provide an output resembling one of the various classes of patterns to which it has been previously programmed. This includes the removal of noise from a corrupted input pattern.

Feature extraction from images (such as line or edge recognition) is showing promise using network approaches.

Control of, for example, robotic and autonomous vehicles deals with optimization in n-dimensional space of an algorithm to produce a desired learned response. A demonstrated example is the problem of keeping a vertical broom handle from falling over due to the force of gravity while control forces

[18] P. K. Simpson, *Foundations of Neural Networks*, in book - *Artificial Neural Networks: Paradigms, Applications, and Hardware Implementations*, edited by E. Sanchez-Sinencio and C. Lau, IEEE Press, NJ (1992). NRL Library Q 335 .S545 (1990).

are applied to the base. Neural network algorithms for learning successful behavior to such problems have been implemented[19].

Function simulation is a method of fitting a desired behavior with a functional dependence which properly reproduces that behavior. Control of robotics falls in this category, although robotic behavior is combined with "learning" to obtain a desired functional dependence which may not be formulated or understood. In general, per Cybenko [Footnote 9], any function may be simulated with neural networks given enough parameters, just as any function may be simulated with higher order polynomials. For some applications, the convenience of neural networks may provide some advantages for simulation.

These applications have been briefly listed to introduce the most frequently encountered categories of existing programs. Clearly, a number of additional problems may be successfully investigated using these techniques. It remains to see how innovation and imagination will implement these challenges productively.

[19] D. W. White and D. A. Sofge, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold, NY (1992). NRL Library TJ 217.5 .H36.

# 4. NRL PROGRAMS RELATED TO NEURAL NETWORKS

## 4.1. ADVANCES IN FUNDAMENTALS OF NEURAL NETWORKS

A number of programmatic efforts at NRL have been directed at improving the basic algorithmic approach to certain types of neural networks. A larger number of programs take advantage of the enhanced values such networks offer for specific applications. These programs are discussed briefly below.

### 4.1.1. Classical Optimization using Clustering

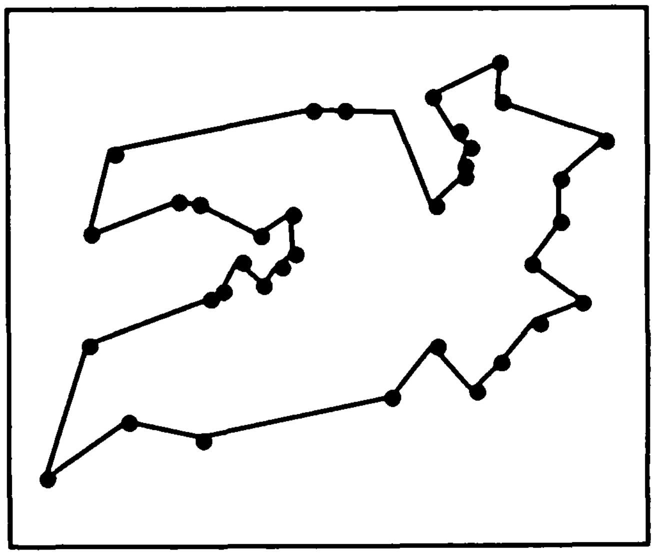
Combinatorial optimization problems (e.g. traveling salesman problem, TSP, clustering, and task assignment) arise in many fields. The number of possible solutions in such problems is so large that finding the best solution is beyond exhaustive search; hence, in practice one is content with finding a good solution. The Hopfield model of neural networks provides an attractive approach. At NRL the challenge of clustering using neural networks has been addressed[20] as it is basic to several programs involving classification and identification of signals; clustering is also of interest in other fields involving image analysis[21], taxonomy, etc.

The general clustering procedure involves partitioning a set of N patterns into K clusters such that the elements in each cluster have a greater similarity to each other than to the elements in the remaining clusters. Formulating clustering (or other problems) in terms of a Hopfield network involves embedding a discrete problem in a continuous domain; therefore, one must construct an objective function which contains the cost as well as constraints [Footnote 20]:

20 B. Kamgar-Parsi and B. Kamgar-Parsi, *Hopfield model and optimization problems*, in *Neural Networks for Perception*, edited by H. Wechsler, 2, 94-110, Academic Press (1992).

21 B. Kamgar-Parsi, B. Kamgar-Parsi, and H. Wechsler, *Simultaneous fitting of several planes to point sets using neural networks*, Computer Vision, Graphics, and Image Processing, 52, 341-359 (1990).

$$E = (A/2)\Sigma_i \Sigma_p \Sigma_{q \neq p} V_{pi} V_{qi} + (B/2)\Sigma_i (\Sigma_p V_{pi} - 1)^2 + (C/2)\Sigma_i \Sigma_p d(x_i, y_p) V_{pi}^2$$



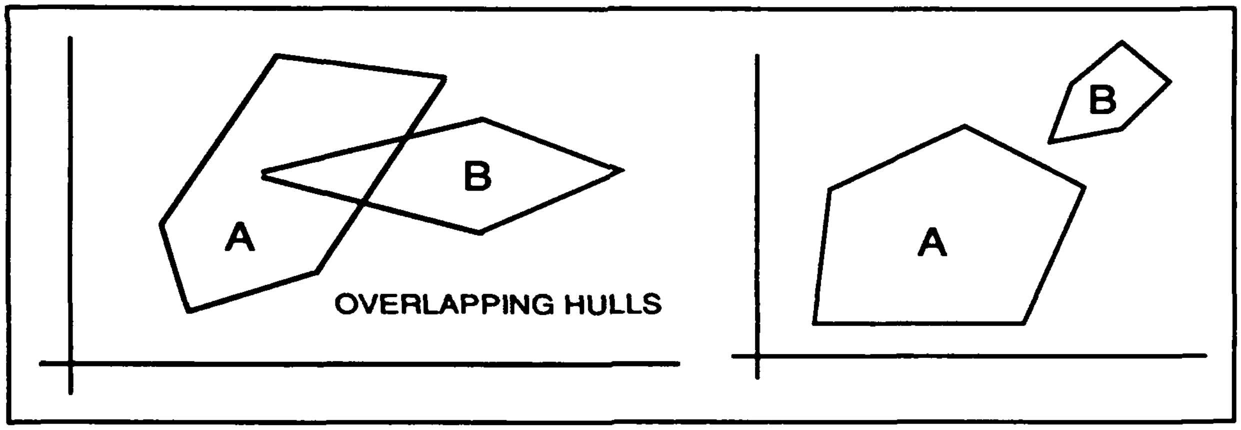
where the sum over i encompasses the N patterns, and the sums over p and q are over K clusters. The neuron activity, $0 \leq V_{pi} \leq 1$, represents the strength of attachment of pattern $i$ to cluster $p$. Here, the $C$-term is the cost, and the $A$ and $B$ terms enforce the constraint that, at the end of the search, each pattern must uniquely belong to one and only one cluster. The network dynamics are given by the set of ordinary, nonlinear differential equations $db_{pi}/dt = -\partial E/\partial V_{pi}$, where $b_{pi}$ are the neuronal input potentials and are related to the firing rates by a sigmoid function. Tests have shown that this neural network approach performs better than conventional techniques for solving clustering problems.

Since constraints are included in the objective function, the network may, in some of its trials, find meaningless or invalid solutions (i.e. solutions that do not satisfy the constraints or the "syntax"). For clustering, this is a manageable difficulty, while, for TSP, it is severe. Hopfield and Tank, in their seminal paper, reported great success in solving the TSP. Their results were subsequently challenged by researchers who could rarely find valid solutions; thus this entire approach became suspect. However, it was found at NRL [Footnote 20] that the difficulty lies in the use of an overly complex TSP syntax, rather than in the neural network model; this was evidenced by clustering, which has a simpler syntax and performs well. It was also found at NRL that dynamical stability analysis of solutions, a seemingly intractable undertaking, reduces to only a few inequalities which may be useful in two respects: (i) selecting optimal values for network parameters, and (ii) selecting the most appropriate formulation in cases where constraints may be formulated in more than one way. Furthermore, it is shown that by analyzing the eigenmodes of the connection matrix, a heuristic formulation may be revised and made more effective[22].

### 4.1.2. Generalization and Learning by Convex Topology

Consider a simple perceptron mapping an input space $\Omega_x$ into $\Omega_y$. If a distinct class coding is forced upon $\Omega_y$, then the perceptron's discrimination capacity is limited to linearly separable regions in $\Omega_x$. Equivalently, a ***necessary*** condition for zero error of the transformation of $\Omega_x$ into $\Omega_y$ is that the class specific convex hulls (defined in next paragraph) must be disjoint (not overlap) in $\Omega_x$[23].

The convex hull of a set of sample points in n-dimensional space is the minimum surface that boun'ls the points. A physical example in two-dimensional space is to imagine an elastic membrane that is stretched and placed around a set of points. If the membrane is released, it shrinks and surrounds all the points. The convex hull is described by the vertices, the points that deflect the membrane.



[22] B. Kamgar-Parsi and B. Kamgar-Parsi, *A revised clustering technique using a Hopfield network,* Proc. World Congress on Neural Networks (WCNN '93), Portland, OR, to appear July, (1993).

[23] J. Willey, et. al., *Generalization and Learning by Convex Topology,* Proceedings of the IJCNN '92, Beijing, China, Nov. 3-6, Vol. II, pp 395-401 (1992).

## NRL PROGRAMS RELATED TO NEURAL NETWORKS

The necessary conditions of disjoint convex hulls form the basis for a new multiple layer perceptron (MLP) learning algorithm. For a specified class coding in the output space $\Omega_n$ of an MLP network, a set of topological constraints is generated in $\Omega_{n-1}$, the space sampled by the perceptrons forming $\Omega_n$. A closed form cost function was developed to measure the disjointness of the class specific convex hulls in $\Omega_{n-1}$. Cauchy simulated annealing is then used to perturb the weights of the perceptrons forming $\Omega_{n-1}$, to minimize the objective function, and to satisfy the topological constraints. The convex hull locations in $\Omega_{n-1}$ also "code" $\Omega_{n-1}$. Hence, the algorithm iterates backwards through each layer of the MLP network. Once the necessary conditions have been satisfied, gradient descent of the weights between $\Omega_{n-1}$ and $\Omega_n$ completes the desired mapping.

Bounds on the probability of error of trained MLP's may be understood from the topological perspective. In any layer space $\Omega_z$, if the class specific convex hulls intersect, then a lower bound on the probability of error has the form

$$P_r(\text{Error}) \;\alpha\; \Sigma_i P(i) \int p_z(z|i)dz$$

where P(i) is the *a priori* probability of the class i, $p_z(z|i)$ is the probability density function in z of class i, and the integral is over the volume(s) in z in which two or more class specific convex hulls intersect. Similarly, an upper bound on the probability of error for disjoint convex hulls in $\Omega$, is:

$$P_r(\text{Error}) \geq 1 - \Sigma_i P(i) \int p_z(z|i)dz$$

where the integration in this case is over the convex hull volume of the class i samples in $\Omega_z$.

In conclusion, this algorithm demonstrates the *necessary* topological constraints which an MLP network must satisfy. Furthermore, the constraints alone may form the basis of a stochastic learning algorithm for MLP's. Bounds on the probability of error may also be expressed in terms of the convex hull locations in each layer space. This demonstrates the learning algorithm and generalization properties for an MLP having two layers of weights capable of solving the the XOR case.

### 4.1.3. Collective Recall for Classification via a Relaxation Neural Network

A method has been introduced[24] which determines the classification of a measurement by comparison with the distances to points in feature space of known classification. Unlike nearest neighbor recall which generates jagged (spurious) decision boundaries, this method provides smooth decision surfaces by taking into account a number of identified positions for this process.

Consider a number of points in n-dimensional space representing successive measurements of parameters associated with either class A or class B. The patterns of the classes are assumed to be distinguishable. Hence, any measurement of parameters associated with an unknown class belonging to either A or B may be characterized by comparison of the points associated with the unknown relative to the two characterized clusters.

A useful algorithm for this purpose is one which describes the evolutionary behavior of neuron values according to the relationship

$$b_i(t+1) = f[b_i(t) + \Sigma_j w_{ij} b_j(t)]$$

where the function f[ ] is

$$f[\ ] = \left\{ \begin{array}{l} 1,\ x>0 \\ x,\ -1<x<1 \\ -1,\ x<-1 \end{array} \right\}$$

and the weights are chosen such that

$$w_{ij} = \left\{ \begin{array}{ll} e & \text{if i,j in same class} \\ -f & \text{if i,j in different classes} \end{array} \right.$$

24 A. Schultz *Collective Recall via the Brain-State-in-a-Box Network*, to be published.

This equation, upon iteration over time, converges to values of $b_i = +1$ or $-1$, thus forcing a classification based on the interrelationships given by the weights $w_{ij}$. An initial set of $b_i(t=0)$ values are chosen according to the relationship

$$b_i(t=0) = \exp(-|x_i - z|^2/d_o)$$

where $|x_i - z|$ is the distance between the point to be classified and the existing points with identified classification, and $d_o$ is a scale parameter.

Alternative assumptions have also been examined, along with an analysis of the values from the point of view of the principal axis system. This method is compared with other methods and the merits of each are discussed.

## 4.2. APPLICATIONS OF NEURAL NETWORKS

### 4.2.1. Adaptive Retina-Like Processing in Detector Arrays

Image formation with infra-red focal plane arrays is plagued with irregularities (typically ±10%) in the response of each pixel, which vary widely in sensitivity. These same irregularities exist with the human retina, although photobleaching and "signal processing" involved in human vision are able to adapt to such variations. The similarity between the algorithms involved with neural networks and those associated with image processing has been discussed[25]. Self-adaptive algorithms for IR focal plane arrays have been examined and introduced at NRL to obtain a significantly improved image using neural network approaches[26].

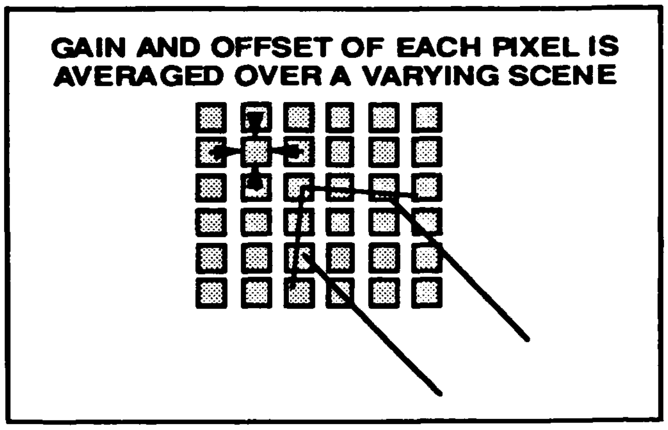
A linear nonuniformity correction (NUC) is of the general form

$$y_{ij} = G_{ij} x_{ij}(\Phi_{ij}) + O_{ij}$$

25 J. Y. Jau, Y. Fainman, and S. H. Lee, *Comparison of artificial neural networks with pattern recognition and image processing*, Appl. Optics **28**, 302-305 (1989).

26 D. A. Scribner, J. T. Caulfield, K. A. Sarkady, and M. R. Kruer, *Real-time Implementation of Adaptive Nonuniformity Correction for IR Focal Plane Arrays*, Aug. 12, 1992.

where $y_{ij}$ is the corrected output voltage from a pixel in the i-th row and j-th column, $x_{ij}(\Phi_{ij})$ is the voltage from the i,j-th pixel, $\Phi_{ij}(t)$ is the time-dependent image of irradiant photon flux on the detector, $G_{ij}$ is the gain, and $O_{ij}$ is the offset correction. This technique suffers from pixel nonlinearities and instabilities, which limit the accuracy of the calibration.


GAIN AND OFFSET OF EACH PIXEL IS AVERAGED OVER A VARYING SCENE

An adaptive NUC studied at NRL has been applied to an image moving across the pixels. A local spatial neighborhood average $f_{ij} = (1/4)*(y_{i+1,j} + y_{i-1,j} + y_{i,j+1} + y_{i,j-1})$ is then used to estimate the desired average output of pixel ij. Using an adaptive LMS approach, a neural network algorithm has been derived which gives a recursive relationship between the n-th and (n+1)-st gain and offset for each frame[27] (each element except $\alpha$ has the subscripts ij):

$$G_{n+1} = G_n - 2\alpha x(y-f)$$

$$O_{n+1} = O_n - 2\alpha(y-f)$$

where $\alpha$ is chosen to control the step size and consequently the stability. These equations were derived using the method of steepest decent to converge to the appropriate values. To prevent drifting of the gain values toward zero, a frame-by-frame normalization of G and O is performed.

This algorithm dramatically improves the quality of images from IR focal plane arrays, successfully compensating to a high degree for irregularities in pixel area and gain. It is being adopted in systems for use within DoD, and has applications in many other areas involving the calibration of detector arrays.

[27] D. A. Scribner, K. A. Sarkady, M. R. Kruer, J. T. Caulfield, J. D. Hunt, and C. Herman, *Adaptive Nonuniformity Correction in IR Focal Plane Arrays*, SPIE **1541**, 100 (1991).

### 4.2.2. Sensing of Image Motion with Gabor Receptive Fields

The architecture of the motion sensing system is inspired by neurophysiology research on the primate vision system in cortical processes. In the primate vision system, motion is sensed by magno cells in the visual pathway.

The model of the magno receptive fields is derived from the functionalities of the simple cells, complex cells and hypercomplex cells in the visual cortex. Receptive fields of simple cells can be modelled as 2-D Gabor function, i.e. a Gaussian modulated sinusoidal function:

$$G(x,y) = \exp\ \{-[x^2/(2\sigma_x^2)+y^2/(2\sigma_y^2)]\}\ \exp\ [-i\omega_o(x\cos\theta + y\sin\theta]$$

This function is an ellipsoidal Gaussian modulated at angular frequency $\omega_o$ with an inclination $\theta$ relative to the axes. The complex cell, modelled by a quadratic combination of the outputs from even and odd simple cells, is responsible for energy measurement. The hypercomplex cell, modelled by spatial differentials of the Gabor function, measures the change of spatially oriented contrasts.

The motion sensing system calculates local motion information and performs Gabor and differential Gabor transformations at each location of the image. It computes motion accurately without doing global smoothing. This algorithm relies only on a single "expanded frame", i.e. single image frame accompanied with its time derivative.

The essential part of computational model is the following. A moving small image patch can be described as $I(x',y',t) = I(x-u_xt,y-u_yt)$, where $V = (u_x,u_y)$ is the motion velocity vector of the patch. Let $I(x,y)$ be the image frame sampled at time $t=0$, and $J(x,y) = dI(x,y,t)/dt|_{t=0}$ be the time derivative of the time-varying imagery sampled at the same time t. Applying the Gabor receptive fields to $J(x,y)$, we have[28]:

[28] T. R. Tsao and V. C. Chen, *A Neural Network for Optical Flow Computation based on Gabor Filters and Generalized Gradient Method*, International Journal of Neuro-Computing, No. 5, pp. 1-21 (1993).

$$\iint J(x,y)G(x_0-x,y_0-y)dxdy = - V\cdot\iint I(x,y)\nabla G(x_0-x,y_0-y)dxdy$$

To fully constrain visual motion, a collection of Gabor receptive fields, different only in orientation, are needed to determine the two-dimensional motion vectors. Then, the local motion is extracted by minimizing the Least Square Error function.

This algorithm overcomes the limitations and the complexity of other models by taking time derivatives of the Gabor responses as a carrier of motion information and by introducing a concept of differential Gabor receptive fields to avoid spatial derivatives of image intensities. Therefore, the system does not require pre-smoothing of images and extracts motion information from only a single expanded frame or two successive frames.

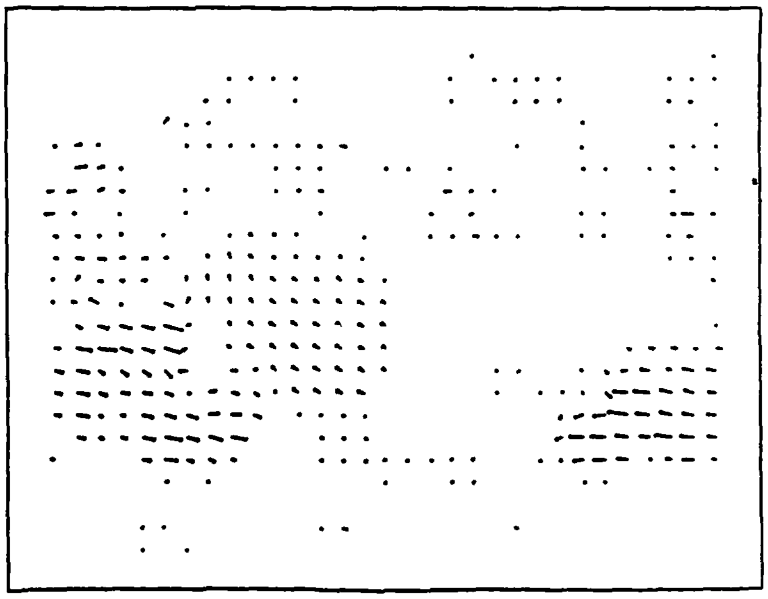
Preliminary research and computational experiments on motion sensing have been conducted on conventional computer. Gabor pyramid was used for extracting two-dimensional velocity flow fields (image flow) of a non-homogeneous motion. Motion information was integrated from multiple levels by a coarse-to-fine control strategy and combined to obtain the image flow[29].

To illustrate the effectiveness of the Gabor pyramid for extracting image flow, an image sequence (called Hamburg taxi in the accompanying figure) containing non-homogeneous motion was selected for the study. Only two frames from the image sequence are used. The Gabor motion vector contains 4 orientations. There are 4 moving objects in the Hamburg taxi sequence. The taxi is turning right at the corner with velocity about 1.0 pixel/frame. A car in the lower left is moving from left to right with velocity about 3.0 pixel/frame. A van in the lower right is driving right to left with velocity about 3.0 pixel/frame. There is a pedestrian in the upper left with velocity about 0.3 pixel/frame. The image flow computed from two frames of the sequence is shown in the figure labed "The image flow ...." Our result shows that the performance of the Gabor pyramid is very good. The image flow for the two fast moving objects (a car and a van) and for the normal moving taxi is

[29] V. C. Chen and T. R. Tsao, *Gabor-Wavelet Pyramid for the Extraction of Image Flow*, Proceedings of 1993 SPIE Mathematical Imaging: Wavelet Applications in Signal and Image Processing (1993).

correctly extracted. The slow moving pedestrian in the upper left can also be seen.

Hamburg Taxi sequence



The image flow for Hamburg Taxi sequence

### 4.2.3. Polynomial Neural Networks for Signal/Image Processing

A method for obtaining a least squares solution to a high-order polynomial with several variables was introduced in a novel manner by A. G. Ivakhnenko[30] and is known as the Group Method of Data Handling (GMDH). The GMDH algorithm provides an approximate solution of the complete Kolmogorov-Gabor polynomial, which may be written as

$$y = a + \Sigma_i b_i x_i + \Sigma_{i,j} c_{ij} x_i x_j + \Sigma_{i,j,k} d_{ijk} x_i x_j x_k + \ \ldots$$

In computer implementations of the GMDH algorithm, the Kolmogorov-Gabor polynomial is replaced by a composition of lower order polynomials. This composition results in a multi-layer feedforward polynomial network which, for example, may be represented by third order polynomial transfer functions $\boldsymbol{S}$, $\boldsymbol{D}$, $\boldsymbol{T}$ , where

$$\begin{aligned}
\boldsymbol{S} &= u_0 + u_1 x_1 + u_2 x_1^2 + u_3 x_1^3 \\
\boldsymbol{D} &= v_0 + v_1 x_1 + v_2 x_2 + v_3 x_1^2 + v_4 x_2^2 + v_5 x_1 x_2 + v_6 x_1^3 + v_7 x_2^3 \\
\boldsymbol{T} &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_1^2 + w_5 x_2^2 + w_6 x_3^2 + w_7 x_1 x_2 + w_8 x_1 x_3 \\
&\quad + w_9 x_2 x_3 + w_{10} x_1 x_2 x_3 + w_{11} x_1^3 + w_{12} x_2^3 + w_{13} x_3^3
\end{aligned}$$

There is a significant difference between the GMDH method and the straightforward method of directly utilizing this expansion. The process of iteratively selecting only the "best" variables y (and successive approximations) prunes the totality of coefficients in the general expansion such that only selected cross terms survive the GMDH treatment. Selecting only those which are "best" (i.e. retaining those values of y for which the mean square error is small) retains only those basic variables x which are best at fitting the observed data.

In an NRL study, using this algorithmic approach, a pulse signal

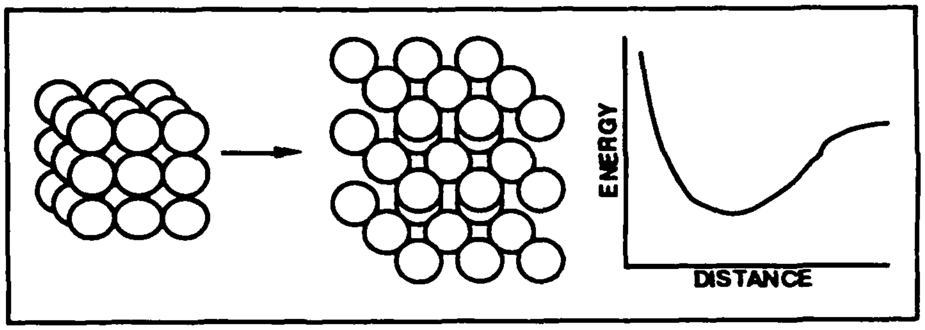
[30] S. J. Farlow, *Self-Organizing Methods in Modeling: GMDH-Type Algorithms*, Marcel Dekker, Inc., NY (1984).

embedded in a chaotic background and an image of ocean waves were treated[31,32]. These examples are typical of the detection of threshold signals or images in a chaotic background (chaotic backgrounds are quasi-deterministic, not random noise). First, the characteristics of the chaotic background are used to train a polynomial neural network as a global predictor. A small coherent signal may then be detected by subtracting the predicted chaotic background from the observed signal plus background. This method appears to give good results in preliminary tests.

This approach is not strictly a neural network approach; however, due to the resemblance between networks and the concepts here, where the results of certain operations are fed into successive iterations, it has been included. It appears to be a powerful technique for fitting complex interrelationships in some cases.

### 4.2.4. Parametrizing Total Energy Databases

The utility of neural networks for approximating complex functions at times makes them attractive as an alternative to other functions used in more standard calculations. One NRL effort involves exploring the prediction of material properties with neural network methods.



Standard formulations for calculating thermodynamic properties and/or material changes from first principles involve the solution of the quantum mechanical Hamiltonian. Accuracies of approximately kT (or about .025 eV)

[31] S. Gardner, *Polynomial Neural Nets for Signal Detection in Chaotic Backgrounds*, Proceedings, Southcon/92, Orlando, Florida, March 10-12 (1992).

[32] S. Gardner, *Polynomial Neural Nets for Signal and Image Processing in Chaotic Backgrounds*. SPIE 1567, Applications of Digital Image Processing XIV (1991).

are required for a reasonable thermodynamic predictions. *Ab initio* calculations are accurate but use excessive machine time; the upper limit to the number of atoms reasonably handled with these calculations is approximately 10. This number is too small to predict the necessary properties of larger clusters which approximate matter more completely.

An initial approach is to use fairly simple formulations of total energy for structures of atom clusters to determine whether the neural network can give an accurate replication of the material behavior predicted with these simple models. If that is the case, subsequent efforts will focus on adjusting the parameters of suitable networks while using *ab initio* calculations for comparison. This parameter adjustment will terminate with cluster sizes of about 10 atoms. Once the parametrization has been introduced through the comparisons of small cluster sizes, the validity of predictions for larger cluster sizes will be examined.

The basic relationships initially used include the Leonard-Jones potential (a classical model involving only the distance between two atoms), the embedded atom potential (involving two-body interactions along with a third function involving the energy of "embedding" an atom in a sea of electrons from the other atoms), and the Stillinger-Weber potential (a function of three-body interactions).

The first network used to fit these basic functions was a feed-forward perception with one hidden layer. A hyperbolic tangent was used to "squash" the output of the layers. The total energy of a cluster of atoms was calculated (for various crystal symmetries) with the classical techniques mentioned above as a function of distances. This constituted a database for subsequent parameter adjustment. The cost function was the least mean square (LMS) between the values calculated by the network and those calculated by the classical models.

A number of optimization approaches have been investigated for adjusting the weights. Experience has been gained with back propagation, simulated annealing, and genetic algorithms. It was found that the back propagation method is extremely time consuming and is therefore not recommended. Simulated annealing offered considerable improvement. Genetic algorithms appeared to map out the n-dimensional space rapidly. Subsequent optimization using the conjugate gradient method represented a quick way to find a good minimum initially approximated by the genetic algorithm.

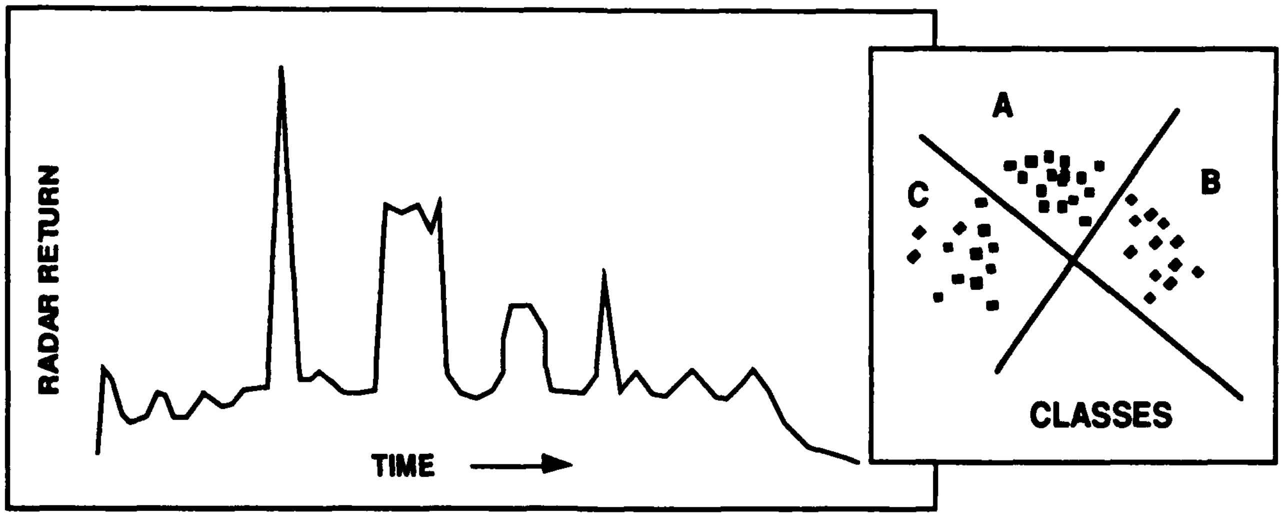
Initial results using classical energy functions show that the neural network is able to simulate the classical behavior to a reasonable degree of approximation (currently about 0.1 eV). Further refinement is underway to determine the true limits of this method.

It should be noted that other approaches to this simulation are to be explored, thus testing the neural network approach against alternatives. In addition, new methods of optimization will be introduced. These methods are outlined in Appendix III.

### 4.2.5. Hierarchical Wavelet Representation of Ship Radar Returns

Short pulse radar returns from ships contain considerable information from which features may be extracted. Such features, if properly interpreted, may be used for ship classification and/or identification. In this NRL effort, an investigation is being undertaken to find efficient representations of large databases of pulsed radar returns in order to economize memory requirements and minimize search time.

Using a model of 400 to 1000 scatterers moving stochastically in response to simulated seawaves, and incorporating such random phenomena as multipath and clutter, signal returns from ships were simulated. The behavior



of these signals was examined for characteristics which may lead to recognition. The wavelet transform was found to be a useful representation of the time-dependent signals from a ship. This transform was used because of its ability to represent impulses characteristic of the sharp features arising from masts and other localized reflectors at certain angles. The basis functions form a complete orthogonal set, useful in representing the signals.

In this investigation the amplitude of the envelope of the extended target signal return was represented by a vector of length 128. This data were then represented by a multiresolution decomposition using a wavelet basis. The multiresolution data were then organized into a hierarchical structure by combining the Linde, Buzo, Gray (LBG) algorithm with a tree-structured vector quantizer (TSVQ) paradigm.

The LBG algorithm is similar to Learning Vector Quantizer (LVQ), a clustering algorithm which converges to the optimal Bayes decision surface. However, in the case of LBG, it is not signal differences but, rather, signal fidelity which is emphasized. The algorithm is initialized by a set of (reference) vectors in n-dimensional space. This n-dimensional space is then partitioned into cells, one cell for each reference vector, defined by all points in the space closest to the given reference vector. The centroid of each cell is calculated based on the available data, providing a different location for a cluster center than initially chosen, whereupon a subsequent choice of vectors is chosen and new cells introduced. This iteration procedure is continued until self consistency is reached.

A TSVQ paradigm quantizes data hierarchically. In the context of the effort described here, a multiresolution representation of the data produces representations of the ship signal return in a number of resolution scales. LBG is repeatedly applied to the coarsest resolution data, adding cells with each iteration until the reduction in distortion falls below a given threshold. At that point, the cell which is the greatest contributor to average distortion is further expanded in the next higher resolution scale by applying LBG to the observations which fall in that cell at the next finer resolution scale. The algorithm proceeds in this way until either the desired distortion level is reached or until the maximum allowable number of cells has been apportioned.

By experimental results, it was shown that the combined algorithm results in data search times are logarithmic in the number of terminal tree nodes, and experience negligible performance degradation (as measured by distortion-entropy curves) from the full search vector quantization. Furthermore, it has been shown that the combined algorithm provides an efficient indexing scheme (with respect to variations in aspect, elevation and pulse width) for radar data; this is equivalent to forming a multiresolution aspect graph or a reduced target model.

### 4.2.6. Inverse Synthetic Aperture Radar (ISAR) Signal Classification

ISAR signals are used to classify ships based on the images formed by the radar process. These images are distorted compared with those recognized by ordinary visual observation, and experienced operators are required in order to extract sufficient detail to make accurate classifications from these images. This research program is based on the exceptional utility anticipated from an automatic recognition algorithm.

The classification of simulated ISAR images using neural networks is described in recent papers[33,34,35]. The simulated image is projected onto the horizontal range axis using a measurement of the weighted variance in the image:

$$a_i = [\Sigma_j(j-r_i)^2\rho_{ij}]/K$$

where $\rho_{ij}$ is the intensity in simulated range-doppler cell i,j, and $r_i = \Sigma_j j\rho_{ij}$ is the center of mass in range bin i. $\rho_{ij}$ is normalized so that $\Sigma_j\rho_{ij} = 1$; an overall

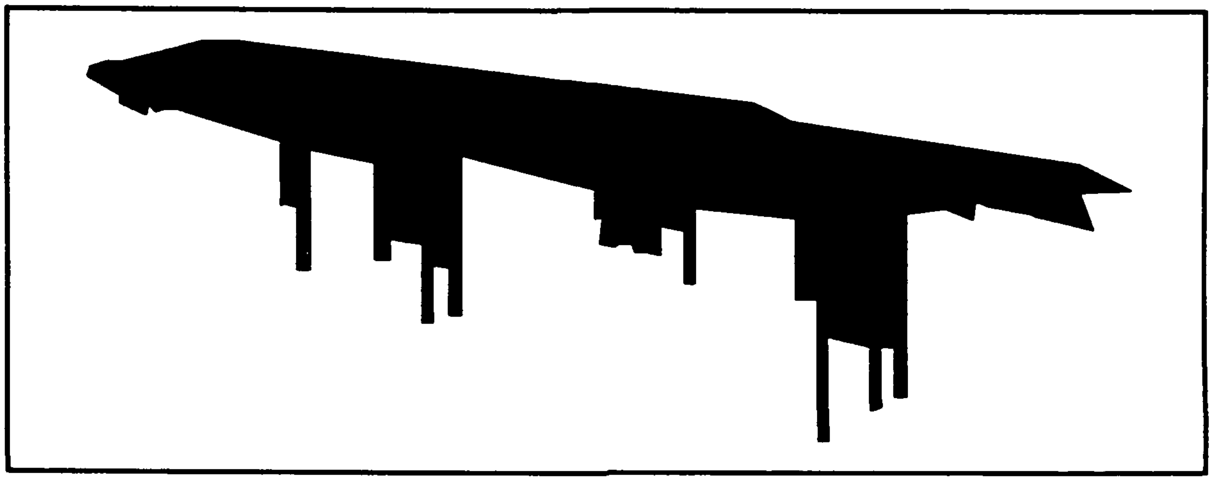
33 C. M. Bachmann, S. A. Musman, D. Luong, and A. Schultz, *Unsupervised BCM Projection Pursuit Algorithms for Classification of Simulated Radar Presentations*, accepted for publication in Neural Networks.

34 C. M. Bachmann and D. Luong, *Extensions of Unsupervised BCM Projection Pursuit: Recurrent and Differential Models for Time-Dependent Classification*, to appear in Proceedings of the 1993 World Congress on Neural Networks, Portland, OR, July 11-15 (1993).

35 C. M. Bachmann, S. A. Musman, and A. Schultz, *Classification of Simulated Radar Imagery using Lateral Inhibition Neural Networks, Neural Networks for Signal Processing, II* - Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing, Copenhagen, Denmark, Aug. 31-Sept. 2 (1992).

normalization constant, K, takes account of the variation across all range bins, and is invariant to shifts in scale in the simulated Doppler dimension.

These range-bin moments become the input to the neural network responsible for identifying the Perceptual Class of the simulated ISAR presentations. Thus, not all of the image information is presented to the network; however, the preprocessing step does take advantage of underlying symmetries in the imagery; features may change in the simulated Doppler domain but they will remain constant in the simulated range domain.



Two approaches in the design of the neural network were tested for this recognition/classification problem: 1) the usual back-propagation model, and 2) the unsupervised Projection Pursuit learning algorithm of Bienenstock, Cooper & Munro (BCM). In the BCM theory, an alternative cost function is used[36]:

$$E = -\{(s^{(n)}_{i\mu})^3/3 - \gamma(s^{(n)}_{i\mu})^2 \mathcal{E}[(s^{(n)}_{i\mu})^2]/4\}$$

where $\mathcal{E}[\ ]$ is the average over a suitable time interval of the enclosed quantity. The $s^{(n)}_{i\mu}$ are the values at the $i,\mu$-th node in layer n, and $\gamma$ is an empirically determined scale factor adjusted to keep the network from oscillating, ensures good convergence, and preserves the units of the equation. Minimization of this function favors directions of statistical skewness (i.e. directions where the projected distribution is bi-modal), thus revealing the inherent structure in the

[36] N. Intrator and L. N. Cooper, *Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions*, Neural Networks, 5, 3-17 (1992).

data. This approach is also used to introduce stability into the search process through the dynamics of the quantity $\mathcal{E}[(s^{(n)}{}_{i\mu}{}^2)]$. In the BCM theory, "cells" are typically coupled using lateral inhibition:

$$s^{(n)}{}_{i\mu} = f(b^{(n)}{}_{i\mu} - \alpha \Sigma_{k \neq i} s^{(n)}{}_{i\mu})$$

where $\alpha$ is an empirically determined "inhibition" factor, and the $b^{(n)}{}_{i\mu}$ are the direct output values from the indicated layer. Coupling in this manner may be used to remove redundancies in the weights of the various cells in the network. The coupling also means that individual cells may now find directions which may have either bi-modal or multi-modal projection distributions.

In the paper by Bachmann, Musman, Luong, and Schultz[37] a database of some 4896 simulated ISAR presentations of 21 different ships was used; the database was partitioned into three sets: one for training, one for cross-validating during training, and one for evaluating generalization after training. The best performance for the task of distinguishing simulated combatants from simulated commercials was obtained by BCM, ~ 78%. Most striking, however, was that in comparing 60 trials of this approach with 60 trials of the back propagation model, the average rate of recognition of combatant ships for BCM was ~72%, whereas for back propagation it was ~59%. These results were obtained for classification of single frames of simulated ISAR; ongoing work is exploring the use of extensions of the BCM model to obtain target classification from multiple frames (Bachmann and Luong, 1993)[38].

These and related network algorithms have also been used to extract small signals buried in noise, as well as features of Jet Engine Modulation (JEM) signals associated with radar back reflection.

### 4.2.7. Automatic ESM Systems

Automatic ESM systems have been successfully used in the Learning Vector

37 loc. cit.

38 loc. cit.

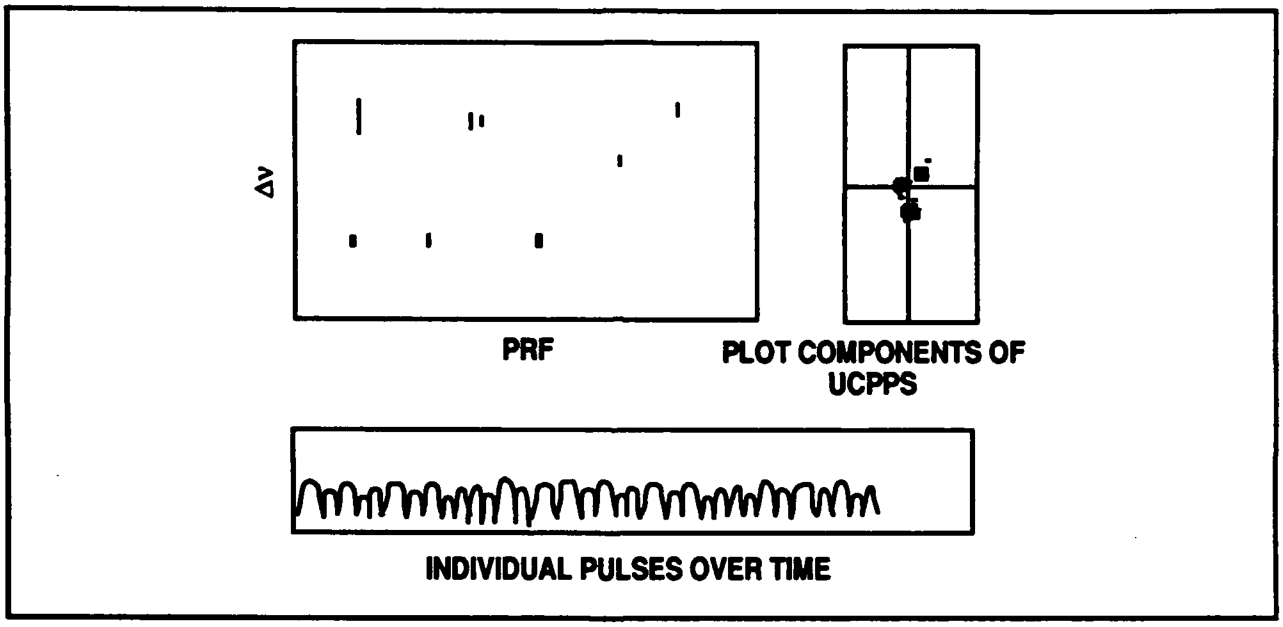
Quantizer (LVQ) algorithm[39] for the characterization of radar signals at NRL. The LVQ algorithm is a major variant of the Self-Organizing or Topology-Preserving Map algorithm of Teuvo Kohonen developed in the early 1980's for speech recognition application. As implemented here, the algorithm acts as an adaptive clustering method operating in an unsupervised learning mode. The algorithm uses for input a set of 15 unconventional pulse parameters (UCPPs). Each parameter of the set is treated as a component of a 15-dimensional vector.

Conceptually, the LVQ algorithm is based upon closest matching between a set of class vectors and the presented data vectors. The class vectors are initialized randomly and are typically of a uniform small length near the origin. Effectively, only the direction in space is randomized. The data vectors may require some transformation or normalization as well. The normalized data vectors are then presented sequentially to each of the class vectors. Whichever class vector is closest as measured by the Euclidean distance (square root of the sum of the squares of the vector component differences) "wins." The winning class vector then has its components modified by the Kohonen learning rule: the new class vector is moved toward the data vector by a fraction of the difference between the old class vector and the data vector. This fraction is known as the "learning rate" and can be related to the number of data vector presentations required for complete learning.

[39] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, New York, (1984).

## NRL PROGRAMS RELATED TO NEURAL NETWORKS

One result of this study was that the optimum learning rate was found to be related to the number of pulses in one to two radar beams. Each data vector has an opportunity to modify any of the class vectors, but only the closest class vector is modified. The result of this presentation of data is that certain class vectors will "grow" in length much more than others, and the class vector direction may change as well. Each one of the group of class vectors which is "growing" eventually comes to rest in the centroid of a unique cloud of pulse parameters identifiable with an emitter. If the data vectors are associated with their closest class vector, these class vectors will then have sorted a set of data vectors (radar pulses) into their respective classes (emitters).



The above algorithm describes initial work with four major applications in mind: (1) control of software bin parameter limits to sort emitters, thereby reducing processing burdens, (2) control of hardware bin parameter limits in an advanced signal processor developed under the 6.2 S&T block, (3) as an "analyst's aid" in the manual analysis of radar collections, and (4) as an automatic software-driven pulse deinterleaver in the front end of an automatic emitter analysis system. All four applications have been demonstrated. Most impressive is the last application which is capable of emulating experienced human analysts to a high degree. When the automatic system and experienced human analyst are given the same data set to classify, the classifications of radar signatures typically match to within a few hundredths of a unit; a match of under 10 units is considered to be close.

The initial work in this area is soon to be published[40]. It describes the early results of a neural network which uses only the UCPP's. However, research has progressed, and the algorithm has grown to take advantage of RF and pulse repetition interval (PRI) characteristics as well as the UCPP's. These parameters and the RF characteristics are used to form initial classes which are then merged when necessary on the basis of similarity of PRI's. The resultant algorithm is quite robust in that shape and density of the clusters have no effect on its performance, and relative cluster separation has only small effects. This algorithm now more resembles the Dynamic Vector Quantizer algorithm of Franck Poirier[41] in that it uses measures of dispersion to dynamically create new class vectors. This overcomes one of the shortcomings of the LVQ in that it must have some upper limit of number of class vectors. Current plans envision this algorithm as the central software element of a stand-alone automatic processing and analysis system.

### 4.2.8. Generalized Probabilistic Neural Networks

Alternative methods of signal processing are important, particularly when the background noise is not Gaussian (which is true in a surprising number of cases). Methods of accounting for Gaussian noise are well understood, and matched filters are often used in such cases.

There are significant opportunities to improve the detection of signals in the presence non-Gaussian noise, and this field is not yet well understood. Neural networks provide an opportunity to develop algorithms for this purpose. The following algorithm has been shown to give 10-15 dB improvement in signal recognition over methods which employ classical linear filters assuming

[40] J.C. Sciortino and J.R. Sevick, *ESM Applications for the Learning Vector Quantizer Algorithm*, NRL Memorandum Report #(TBD) under review at Branch level, Naval Research Laboratory, Washington, DC. (1993).

[41] F. Poirier, *DVQ: Dynamic Vector Quantization - an Incremental LVQ*, pp 1333-36, in Proceedings of the International Conference on Artificial Neural Networks (ICANN-91), Espoo, Finland 24-28 June 1991, edited by T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Elsevier Science Publishers B.V. North Holland, (1991).

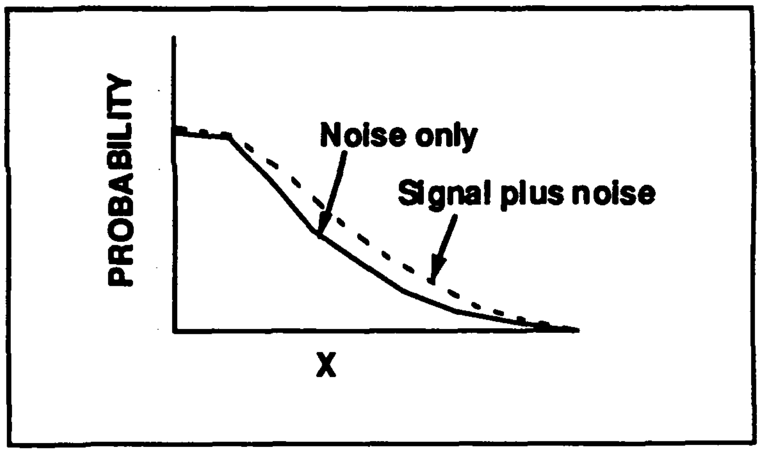
Gaussian noise. Further, these algorithms may be readily introduced with parallel processors, providing additional speed for signal processing hardware.

### 4.2.8.1. Gram-Charlier Neural Network

This method is designated the Gram-Charlier Neural Network (GCNN)[42,43]. A conditional density function g(x|k) is chosen to represent the statistical values of the function being examined:

$$g(x|k) = [(1/(2\pi\sigma^2)]^{1/2}\exp[-(x)^2/(2\sigma^2)][1-\Sigma_j w_{jk}\mathcal{H}_j(x)]$$

where $\mathcal{H}_j(x)$ is the j-th Hermite polynomial, x is the value of a measured function, and k is the class of a specified signal. The selection of Hermite polynomials forms a complete orthogonal set; the coefficients $w_{jk}$ are determined by the learning algorithm introduced. For a class consisting entirely of Gaussian noise, all coefficients $w_{jk}$ are zero. This function thus serves as a modified Gaussian with which to recognize a given class k for a given signal based on the class of weights to which it belongs. The classification is based on the weights which most closely resemble those obtained by the training procedure.



The training procedure of the GCNN is accomplished in a manner resembling a modified Kohonen learning scheme. Initially, a pretraining procedure utilizes experimentally determined density functions for each class k to calculate the coefficients $w_{jk}^{(\mathrm{training})}$; Classical techniques are used to

[42] M. W. Kim and M. Arozullah, *Generalized Probabilistic Neural Network-Based Classifiers*, IEEE Joint Conference on Neural Networks, Baltimore, MD, June (1992).

[43] M. W. Kim, *Neural Network-Based Optimum Target Detection in Non-Gaussian Noise Environments*, Ph. D. Thesis, Catholic University of America (1992).

determine the coefficients of a complete orthogonal set. A set of random vectors is chosen as the initial set of $w_{jk}$. These vectors are first compared with the values of $w_{jk}^{(training)}$:

The minimum,

$$\text{MIN}\left[\Sigma_j(w_{jk}^{(training)} - w_{jk})^2\right],\ k = 1,2,\ldots n,$$

is found for each random vector of length n having a component index j. For the minimum distance found, say this is for k = g, the existing values of $w_{jk}$ ($w_{jk}^{(old)}$) are updated using the following algorithm:

$$w_{jg}^{(new)} = w_{jg}^{(old)} + \alpha(w_{jg}^{(training)} - w_{jg}^{(old)}),\ j = 1,2,\ldots n.$$

The variable $\alpha$ is chosen empirically to optimize the overall rate of learning.

The above steps are repeated for each class of data introduced.

Once the network is trained with the training set, an unknown signal is introduced by finding the minimum distance with the trained vectors $w_{jk}$. The minimum distance is then a measure of the classification for the unknown vector. The value of $w_{jk}$ may be updated in the same manner as that in the training set, thus introducing variations in signal characteristics with time.

### 4.2.8.2. Generalized Probabilistic Neural Networks with Parzen's Windowing Technique

A further refinement to GCNN has been examined, referred to as the Generalized Probabilistic Neural Networks (GPNN). The basis functions formed in this technique are essentially linear combinations of the GCNN functions, using an approach commonly referred to as the Parzen's windowing technique. The overall conditional density function for this approach is

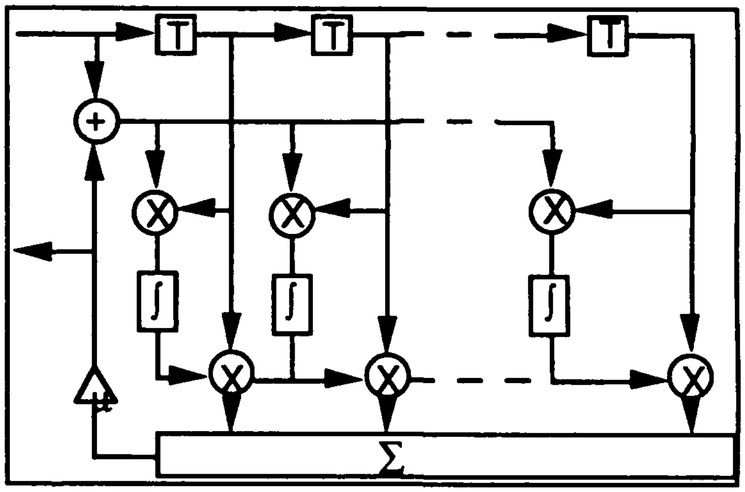
$$g_p(x|k) = (1/N)\Sigma_i g(x-x_i|k).$$

Tests using this function appear to give improved agreement with classification problems of the type described above.

These algorithms have been tested against sample radar signals containing Gaussian and non-Gaussian noise (such as Weibull noise, a common component) and compared with alternative methods such as a Bayesian approach and the usual back propagation methods. They appear to outperform these other techniques by a significant margin.

### 4.2.9. Analog VLSI Artificial Neural Network Research

Algorithm and analog circuit research is currently being performed for three types of analog VLSI artificial neural network analog signal processing circuits. These circuit types include: 1) high frequency continuous-time adaptive learning circuits, 2) programmable analog vector-matrix multiplier circuits, and 3) adaptive learning circuits for infrared detector non-uniformity correction.



The high frequency continuous-time adaptive learning circuit[44] uses continuous-time analog multiplier and integrator circuits to implement a class of gradient decent learning algorithms. Examples of the gradient decent learning algorithms which may be implemented by the continuous-time adaptive learning circuits include the Hebbian, delta-learning rule, and least-mean-square (LMS) learning algorithms with the general form

$$dw_{ij}/dt = -\gamma w_{ij} + \alpha a_i(b_j - \zeta_j)$$

[44] F. J. Kub, E.W. Justh, F. M. Long, and K. K. Moon, *High Frequency Adaptive Learning Element*, GOMAC Conference, Las Vegas (1992).

where $w_{ij}$ is the weight, $\gamma$ and $\alpha$ are constants, $a_i$ is the input, $b_j$ is the output of the linear combiner ($\Sigma_i a_i w_{ij}$), and $\zeta_j$ is the desired value. These circuits have been fabricated using a CMOS foundry process and have achieved 32 dB cancellation of a CW signal at 80 MHz with adaptation times as short as 5 µS. This frequency of operation is greater than two orders-of-magnitude higher than any previous analog circuit implementation. Also, programmable multiple notch filters in which each of the notches has less than 1% notch width have also been demonstrated. The high-frequency adaptive learning circuits may potentially be applied to cancellation of collocated signals in spread spectrum systems, multiple programmable notches for EW systems, adaptive arrays for radar systems, and both passive and active sonar systems. An ONR sponsor contract program is planned to begin in FY94 to investigate the potential performance levels which may be achieved using the continuous-time adaptive circuit approach. The CMOS implementation requires an area of 48 µm x 1.34 mm per learning circuit, allowing up to 200 parallel inputs.

A second class of analog VLSI circuits now under investigation is the programmable analog vector-matrix multiplier circuit[45]. The circuits under development have the potential to perform high speed vector-matrix multiply operations in a small volume with low power dissipation. NRL has designed and fabricated both 32x32 and 128x64 programmable analog vector-matrix multiplier circuits. The 32x32 vector-matrix multiplier circuit has demonstrated full functional capability, while the 128x64 circuit is still under development. The 128x64 vector-matrix multiplier circuit has the potential to perform three billion connections/second. The time required to compute an output vector is approximately 3 µS. The accuracy of these devices approaches seven bits. Potential uses of the 128x64 vector-matrix multiplier array to perform high rate two-dimensional Hartley and Fourier transform operations have also been investigated.

The current research effort to provide infrared detectors with non-uniformity correction involves both an investigation of learning algorithms and potential circuit approaches. The adaptive learning method is the method of steepest descent and its well known special case, the LMS learning method. In the LMS algorithm, the weights are updated proportionally to the product of

[45] F. J. Kub, K. K. Moon, I. A. Mack, and F. M. Long, *Programmable Analog Vector-Matrix Multipliers*, IEEE Journal of Solid-State Circuits, **25**, 207-213, (1990).

respective pixel input and error. By considering only the signs of these factors, various learning method alternatives to the LMS algorithm may be derived. These include the clipped, pilot and zero-forcing algorithms, depending on which factor the sign operation is performed. The trade-offs of these algorithms with those of the LMS algorithm include stability, learning rate, and ease of implementation. The primary advantage of these algorithms is that they allow the use of comparator circuits rather than analog multiplier circuits; this significantly simplifies the analog weight learning circuitry. These algorithms are also compatible with analog integrator implementations which provide long integrator time constants. Simulations have shown good non-uniformity correction for these learning algorithms as well as the LMS when stimulated by a point-source corrupted by random gain and offset noise. The study will continue by modeling the non-idealities associated with circuit implementation of learning algorithms. The neural network correction circuits will implement an appropriate algorithm based on various trade-off issues.

### 4.2.10. Optical Cellular Neural Network

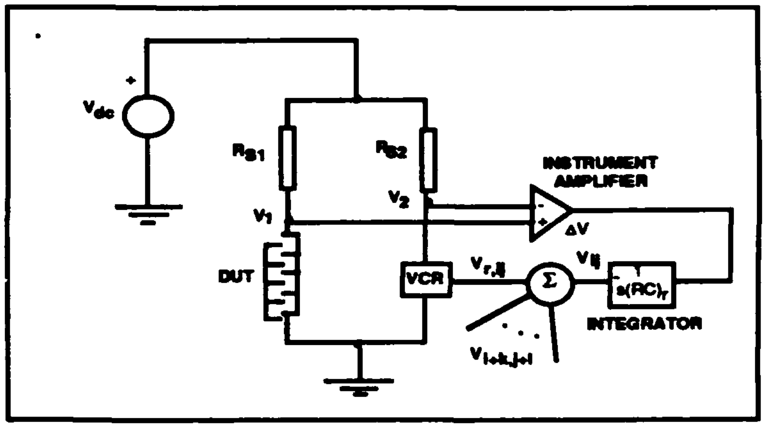
The Center for Bio/Molecular Science and Engineering at NRL has demonstrated the ability to grow neurons in designated high-resolution patterns. This is accomplished using a photochemical process for modifying the substrate on which they grow, and represents an advantage towards investigating (and perhaps utilizing) biological neural networks and their behavior. Among other questions to be pursued by this research include: 1) How do the neurons form synapses? 2) Can the presence of toxins be sensed reliably and inexpensively with such biological networks? 3) Can optical fluorescence of such electrically active cells be used for signal transduction and processing?

As an initial investigation towards that end, NRL research has designed and tested an autonulling DC bridge (ADCB) that is able to detect and amplify picoamp/microvolt signals from nerve cells. A U.S. Patent disclosure for this technology has been filed. Signal filtering through on-site microelectronic processors provides enhanced signals. The ADCB has also been used to detect optical signals; it was then configured in an interconnected optical cellular neural network (OCNN) that can transduce spatial optical signals. Optoelectronics will be fabricated into substrates on which neurons are directly patterned; these devices will also be tested using a projection scheme with the OCNN as well as an electrically sensitive array for spatial detection of electrochemical signals.

## NRL PROGRAMS RELATED TO NEURAL NETWORKS

The nearest-neighbor interconnected neural network utilizing the autonulling bridge as the unit transduction "cell" has been implemented at NRL. An integral feedback system is used to establish a stable null across the bridge; the feedback voltage, V, controls a voltage controlled resistance (VCR) used to correlate V to the input signal being transduced. The stand-alone bridge has been shown to have an asymptotically stable solution. The use of the autonulling bridge as the transducing element has two significant advantages: 1) common noise is reduced by a differential measurement taken across the bridge nodes, and 2) integral feedback reduces high frequency noise on each node. Each of the bridge integrators will have outputs that are governed by the differential equation

$$dV_{ij}/dt = A[V_{1,ij} - V_{2,ij}(V_{ij})]$$



where $V_{1,ij}$ and $V_{2,ij}$ correspond to node voltages 1 and 2, respectively, at position i,j in an array. Signals from an array of transducers suffer from spatial noise related to thermal effects on the transducers and local noise at each transduction point. The nearest-neighbor interconnection emulates a cellular neural network that enables spatial integration. It allows the weighted interconnection of the feedback signals to generate the voltage required to balance each bridge in the network:

$$V_{r,ij} = \Sigma_{k,l}\, w_{kl} V_{i+k,j+l}$$

where $w_{kl}$ are the interconnection weights over each of the neighborhoods. The weights of the interconnections determine the type of spatial filtering that is carried out by the network. Each of the bridges will be at null, so the signal $V_{r,ij}$ that controls the VCR of each bridge will correspond to the input signal on that bridge, but the signal $V_{ij}$ will be the signal corresponding to a spatially filtered output from that point in the array.

## NRL PROGRAMS RELATED TO NEURAL NETWORKS

Conventional techniques for filtering spatial noise was previously done on data after it was collected and stored using a deconvolution operator. The weighted interconnection of the bridge feedback signals generates a two-dimensional pre-processed image. The technology used in this network is compatible with microfabrication and therefore offers itself to miniaturized operation. This configuration is being investigated for use in three situations: 1) for two-dimensional optical transduction that can either be mounted under a substrate properly passivated for cellular deposition; 2) as a focal plane array initially designed to mount on a microscope with patterned neurons registered and focussed onto it; or 3) signals transduced from substrate-based electrodes as the input method in which case a two-dimensional electrical image will be transduced.

The stability of the network has been verified using standard perturbation theory as well as by virtue of a global Lyapunov function. The system equations for the OCNN are similar to a general formulation developed by Cohen and Grossberg of the state equation and associated Lyapunov function of locally interconnected neural networks. The state equations discussed can also be applied to many other paradigms of neural networks (e.g., content addressable memory, short term memory) implying that the interconnected ADCB offers itself to other applications besides the OCNN. Other schemes besides linear weighted sums of the neighbor signals are also feasible yielding a variety of spatial filtering algorithms which may be incorporated into the system.

# 5. ADVANTAGES AND LIMITATIONS OF NEURAL NETWORKS

The "hype" evident in some of the neural network community has tended to obscure some of the true merits as well as the limitations of neural networks. Further, although many of the methods reported by the neural network community are based on sound procedures developed years earlier by the signal processing community, signal processing concepts are often not acknowledged by the neural network community. This sense has been captured in an interesting statement[46]:

> "In one sense neural networks are little more than non-linear regression and allied optimization methods. However, they do have a methodology of their own, which has been developed very rapidly by workers from very diverse backgrounds, most with little or no experience nor training in data analysis. Their pervasiveness means that they can not be ignored. In one way their success is a warning to statisticians who have worked in a simply-structured linear world for too long."

A succinct discussion can guide future attempts by new researchers. Comparison studies are most appropriate; in many cases alternative approaches to solving selected problems have been discussed[47]. Advantages demonstrate utility under the appropriate circumstances. Some of these include

1) Recognition of new relationships - Using methods such as back propagation and other optimization techniques, a number of programs have been shown to give results superior to those obtained from standard signal processing and statistical analyses.

2) High speed treatment of data - Due to the inherent parallel architecture associated with neural network algorithms, it is almost axiomatic that they are suitable for new parallel computational tools now

46 B. D. Ripley, *Statistical Aspects of Neural Networks*, to appear in proceedings of lectures for Seminaire Europeen de Statistique, Sandbjerg, Denmark, 25-30 April (1992).

47 J. P. Ignizio, James, *Alternatives to Neural Networks*, 2nd Government NN Applications Workshop, 10-12 September (1991).

becoming available. Once networks have been "trained," they provide decision tools which are computationally efficient, particularly when used with parallel computer implementation. This provides enhanced speed of data treatment, and should be an advantage in appropriate circumstances.

3) Fault tolerance - The graceful degradation of neural networks under failure of digital memories containing the weights has been noted in some references. This property may, as an alternative, be introduced through conventional techniques such as redundancy. A large number of neural network algorithms require the equivalent of redundancy since a large memory is required to successfully recognize selected patterns or to properly classify signals. Thus, the trade-offs between redundancy with conventional approaches and graceful degradation observed for neural networks will be closely examined during the next few years.

Some of the limitations inherent in the neural network approach include

1) Learning many patterns and "behavior" is difficult, taking large quantities of machine time to obtain tolerable recognition or classification levels. "Experimentation" or trial and error is often necessary to obtain useful recognition behavior. Frequently, large numbers of trials are required for learning. The possibility of converging on false or local minima is a significant problem in high-dimensional space.

2) The interconnection density for large neural network problems requiring global connections is too high for implementation by VLSI technology. This places some limitations on the flexibility of the interconnects and hardware designs possible. Thus, artificial networks (similar to those on biological systems) are restricted to perform computations with local interconnections.

3) Hardware implementation of neural network algorithms may perform in an unstable and chaotic mode, giving unexpected problems not readily expected from experience with traditional digital software.

4) In comparison with memory requirements for standard digital techniques, neural networks require far more memory. For Hopfield or other pattern recognition algorithms, the memory capacity is typically on

the order of $0.15n^2$, where n is the number of memory bits required to remember n bits of a pattern. In contrast, a digital system would have a capacity of $n^2$ bits.

5) Neural networks are limited to single measures of classification fit and do not deal effectively with side conditions; these must be introduced artificially as special algorithmic approaches, if indeed, they can be addressed.

6) In many cases, quantitative performance measures for comparing neural networks against alternative approaches are lacking and remain to be developed.

# APPENDIX I: DEFINITIONS AND TERMS

ACTIVATION FUNCTION - A nonlinear transformation performed at each output of a specified "neuron" prior to connection with another level of neurons. Typical transformations are

$$1/(1+e^{-V_o}) \text{ or } \tanh(\alpha V_z)$$

where $V_0$ is the output "voltage" or value of a neuron. The hard limit of the sigmoid (where the value is 1 if $V_0 > 0.5$ and is 0 if $V_0 < 0.5$) is frequently used to simulate the "firing" or absence of "firing" of a neuron.

ADAPTIVE RESONANCE THEORY (ART) - Proposed by Grossberg, this network switches modes between the learning mode and the stable (classification) mode. It uses nonlinear activation functions and feedback. This type of network activates a new internal node whenever an input pattern is sufficiently different from stored patterns.

BACK PROPAGATION - The process of "training" networks by comparing the output of a network to a desired value and feeding a function of the error into the weights to improve the desired behavior of the network.

FEATURE SPACE - The n-dimensional space conceptually used to represent the various components of a signal or object.

FEEDFORWARD NETWORKS - A network which has no feedback; these may be contrasted with Recurrent Networks.

GABOR FUNCTION - A sinusoidally-modulated Gaussian function used to filter image fields; particularly useful for detection motion in successive images.

HEBBIAN LEARNING - A process by which differences between network output and desired output are used to modify the weights in order to obtain improved agreement between network output and desired values.

## APPENDIX I: DEFINITIONS AND TERMS

HIGHER-ORDER PROCESSING UNIT[a] - A network in which the coupling is through higher order coefficients, such as

$$y = f(\Sigma_j w_j x_j + \Sigma_{j,k} w_{jk} x_j x_k + \Sigma_{j,k,l} w_{jkl} x_j x_k x_l + \ldots)$$

where f() is the nonlinear activation function.

HOPFIELD NETWORK - A network using an iterative algorithm in which the weights are selected from (1,0) patterns, modified for improved "learning;" a "hard-limiting activation function" (1 or 0 for the "neuron" values) is used.

KOHONEN SELF-ORGANIZING NETWORK - A network using unsupervised learning algorithms to extract classes or clusters of information.

LVQ - LEARNING VECTOR QUANTIZER - A Kohonen-type of network starting with a known set of classification vectors; updating is based on classification tests with data.

LYAPUNOV FUNCTION - A "cost" function that decreases (or stays constant) in time as the network evolves. A property of dynamics for the Hopfield Network.

PERCEPTRON - An early artificial neural network consisting of a linear transformation from input to output; this transformation may or may not be followed by an operation such as an "activation function." Linear perceptrons are not even able to solve simple nonlinear problems such as the XOR function and were hence not pursued until the process of implementing non-linear transformations on the product of linear transformations was introduced. The term perceptron is used today to include both linear and nonlinear networks consisting of one or more linear transformation.

RADIAL BASIS FUNCTION - A circle chosen to represent a characteristic value in feature space (used with Reduced Coulomb Energy (RCE) networks). Typically, a classification is "taught" through input features involving independent variables and a known classification. This "maps" feature space according to specific

[a] Y. Shin and J. Ghosh, *The Pi-sigma Network: An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation*, International Joint Conference on Neural Networks, Seattle, WA, July 8-12, pp. I-13-I18, (1991).

classifications learned. Conflict among subsequent radial basis functions introduced is resolved by adjusting the radius of the conflicting functions.

REDUCED COULOMB ENERGY (RCE) - A classifier using radial basis functions or "hyperspheres" to classify feature space. It can form arbitrary decision regions and has been used extensively for classification problems.

RECURRENT NETWORK - A network which incorporates some form of feedback from output "neurons" to input neurons; distinctly different from Feedforward Networks.

SELF-ORGANIZING NEURAL NETWORK - A network in which the weights are adjusted through an algorithm which makes use of data to be classified. The process of classification occurs simultaneously with the adjustment of the weights. The Kohonen network is an example of a self-organizing neural network.

SUPERVISED NEURAL NETWORK - A network in which both input and output data are known, and the objective is to "train" the algorithm to replicate the patterns inherent in the data. Hopfield networks or multilayered networks are examples of supervised neural networks. See Unsupervised Neural Network.

UNSUPERVISED NEURAL NETWORK - A network in which the input data are known, and it is desired to obtain relationships extant in this data. Kohonen networks and ARTnetworks are examples of unsupervised networks. See Supervised Neural Network.

WIDROW-HOFF ALGORITHM - A method of updating the weights by considering each existing pattern and adding a fraction of each new (and properly classified) pattern to the appropriate existing weights to "learn" the appropriate pattern for correct classification.

# APPENDIX II: THE STANDARD LEAST SQUARES APPROACH

The following algorithm is a tried and true method of using the least squares approach to obtain a best fit between experimental data and an assumed function. It has been used for decades and is a powerful technique for many problems. Many "standard" techniques seem to have been overlooked by researchers introduced to the methods of neural networks. This algorithm is introduced here as a convenient reminder for those interested in pursuing neural networks.

This method may be generalized to n-dimensional space, to nonlinear functions, and to still more general problems such as image recognition. This technique is advantageous in that it searches for extrema in a much lower dimensional space than do most neural network approaches. It also has disadvantages: 1) a reasonably close approximation to the function being sought is necessary for the iterative process to converge; 2) it may be more time-consuming than alternative algorithms, particularly when the number of variables becomes large (matrix inversion scales as $n^3$).

Given a function $f_i$ (the subscript i denotes the i-th point in space; the function f is the same for all i)

$$z_i^c = f_i(a_1,a_2,...a_n,x_{1i},x_{2i},...x_{mi}),\ i = 1,2,...n$$

where $z_i^c$ is the i-th calculated value (n parameters), the function consists of variables $a_j$ and m independent variables $x_{ki}$, k=1,...m; this function approximates a set of observed values $z_i^o$. The problem is to adjust the values of the variables $a_j$ to obtain the best fit to the experimentally observed values $z_i^o$ and the calculated values $z_i^c$ using the criterion of minimizing the value of the squared differences (least squares):

$$\text{MIN } [\Sigma_i(z_i^c - z_i^o)^2].$$

The derivatives of the calculated points with respect to each of the variables are defined as

$$d_{ij} = \partial f_i/\partial a_j = [\partial f(a_1,a_2,...a_n,x_{1i},x_{2i},...x_{mi})/\partial a_j]$$

# APPENDIX II: THE STANDARD LEAST SQUARES APPROACH

where $d_{ij}$ is the derivative of the function at the i-th point in space with respect to the j-th parameter. Insert into the expression to be minimized the first two terms of a Taylor's series expansion, and obtain the extremum by setting the derivative to zero (this approximation is exact if higher order terms of the Taylor's expansion do not exist, true if the function is first order in the parameters $a_j$):

$$(\partial/\partial a_k)\Sigma_i[z_i^c + \Sigma_j d_{ij}\Delta a_j - z_i^o]^2 = 0$$

Taking the derivative and shifting the order of the terms in brackets gives

$$2\Sigma_i[\Sigma_j d_{ij}\Delta a_j + z_i^c - z_i^o)]d_{ik} = 0$$

Multiply this expression through by $d_{ik}$; drop the 2, and arrange the terms for matrix multiplication by transposing where appropriate and commuting:

$$\Sigma_i[d^t_{ki}\,\Sigma_j\, d_{ij}\Delta a_j - d^t_{ki}(z_i^o - z_i^c)]= 0$$

where $\mathbf{D^t}$ is the transpose of the matrix $\mathbf{D}$ (capital bold letters represent the full matrix; individual matrix elements are in lower case with subscripts).

Defining $(\Delta z_i) = (z_i^o - z_i^c)$, in terms of matrices, this equation becomes

$$\mathbf{D^tD}(\Delta \mathbf{A}) - \mathbf{D^t}(\Delta \mathbf{Z}) = 0$$

Solving for $(\Delta \mathbf{A})$ (assuming an inverse exists) gives

$$(\Delta \mathbf{A}) = (\mathbf{D^tD})^{-1}\, \mathbf{D^t}(\Delta \mathbf{Z}).$$

This prescription gives an efficient adjustment of the parameters, particularly if the function behavior is nearly quadratic in the function space spanned. This sequence may be iterated to self-consistency by inserting the new set of $a_j$'s and repeating the sequence.

# APPENDIX III. NEURAL NETWORK OPTIMIZATION METHODS

A number of methods are currently used for optimizing complex nonlinear functions in n-dimensional space. An appropriate statement regarding these alternatives appears in a recent workshop [Footnote 47]:

> "In many cases, these new tools have resulted in extremely interesting, and sometimes highly effective approaches - for the solution of important problem types that, for the most part, have resisted solution by more conventional techniques. And thus we certainly have no argument whatsoever as to the need for, and the importance of further research in these areas. However, in other instances, we have noted that AI tools have been, quite simply, inappropriately applied - most often out of ignorance of the existence of more suitable alternatives. And this includes a number of instances of the inappropriate recommendation for, and use of, neural networks."

Several useful nonlinear optimization functions are discussed in this appendix to provide a reference. The discussion here has been restricted to feedforward adaptive networks. The problem of "learning" is that of optimization or minimization of an objective function E(w):

$$E(w) = \Sigma_i[z_i^{nn}(w) - z_i^{o}]^2$$

where the sum is over all training sets (i = 1...m) and the inner difference is between the network output ($z^{nn}$) and the required output observable ($z^{o}$). E(w), of course, is a function of the weights {w}, which are to be found from the minimization procedure. Typical functions employed at each node of the network, as seen in Section 2, include hyperbolic tangents, gaussians, linear ramps, etc. As such, this minimization problem belongs, in the terminology of theoretical computer science, to the class of "NP-complete" problems; that is, to find the absolute minimum of this objective function is more difficult than for a polynomial (of order N) hard. The computer time required to find the solution scales more poorly than a polynomial of degree N. As such, neural network learning belongs to the same class as the classic "travelling salesman" problem.

# APPENDIX III. NEURAL NETWORK OPTIMIZATION METHODS

This issue is revisited at the end of this appendix, but several well-established methods exist to train neural networks:

## 1) Back Propagation

The "standard" optimization method used in the neural network community is back propagation. This method is simply a reformulation of the steepest descent algorithm; as such, it is an inefficient optimization method even though it is an O(N) (order N) method. For reasons that this is the case, see Numerical Recipes[b]. The method is guaranteed (in the limit of small step size - larger step sizes may cause divergences or oscillations) to move the set of weights to the nearest minimum unless so-called "momentum" (i.e. hill-climbing) terms are added to the algorithm to ensure convergence to a global minimum.

## 2) Newton-Raphson

Another method is the non-linear version of the Newton-Raphson method; this is usually used iteratively to find the solution to sets of linear equations. A matrix of second derivatives (derivatives of the objective function with respect to weights i and j) is required. Given a starting guess at the weights, the procedure includes iterative convergence on the nearest minimum using gradient information:

$$\mathbf{W}_{n+1} = \mathbf{W}_n - \mathbf{H}_n^{-1}\mathbf{D}_n$$

where the subscript n implies iteration number, **H** is the Hessian (second derivative) matrix, **W** is the vector of weights, and **D** is the vector of first derivatives. Thus, both back propagation and the Newton-Raphson method generally converge on a local minimum. The disadvantage with the latter is that, although the number of steps required to get to the minimum is, in principle, less than for back propagation, the nonlinear Newton-Raphson method requires a matrix inversion - an O($N^3$) procedure. The second more practical issue is that, in neural networks of any complexity, the Hessian is, in general, pathological. The matrix elements vary over many orders of

[b] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes (The Art of Scientific Computing)*, Cambridge University Press, NY (1986). ISBN 0 521 30811 9

magnitude, and the matrices are ill-conditioned. Numerical precision is often an issue. Put in physical terms, this means that the valleys in the "energy" landscape are very steep, and any small imprecision in their description leads to errors in attempting to follow the valley floor to the minimum.

Even quasiNewton techniques, such as the popular Broyden, Fletcher, Goldfarb, and Shanno (BFGS) method[c], which does not require inverting the Hessian matrix, may not perform satisfactorily due to the usually huge number of variables (weights).

### 3) Conjugate Gradient

Related to the Newton-Raphson method but computationally more efficient is the conjugate gradient method[d] [also Footnote b]. Here, as above, the objective function E(w) is approximated by a quadratic form in the weights; but because of a mathematical "trick" (see reference b) only gradient information, not second derivative information, is needed. Migration from an assumed point in function space initially proceeds along the negative of the gradient of the function. A series of line minimizations are performed along so-called "conjugate directions," and the total minimization is performed in $O(N^2)$ operations. Direct calculation of the Hessian is bypassed. Again, the method descends to a minimum near the starting point. Many different starting points are frequently attempted to find a good minimum. The method is probably more stable than the Newton-Raphson method above because direct matrix manipulation is not performed.

### 4) Simulated Annealing

Given the foregoing, it is clear that some "intelligence" is needed in the choice of where to focus attention in the configuration space of all the weights. This may be done via the method of simulated annealing. Here, the analogy of minimization may be compared with cooling a liquid until it crystallizes. This

[c] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, NY (1981). NRL Library QA 402.5 .G54

[d] C. Charalambous, *Conjugate gradient algorithm for efficient training of artificial neural networks*, IEE Proceedings **139**, 301-311 (1992).

is analogous to mathematically simulating a high-energy disorded system to a cooler ordered state. Thus, since the minimization problem is analogous to a thermodynamic problem, standard techniques of numerical statistical mechanics may then be applied. These techniques include Monte Carlo or molecular dynamics. When Monte Carlo techniques are employed, the system of weights is assigned a "temperature." Small changes in weights are made, the objective function E(w) is evaluated, and the change is adjusted according to the Boltzmann factor, $e^{-\Delta E/kT}$. In the case of molecular dynamics, the weights are assigned a fictitious velocity and are updated in time following the Hamiltonian (comprising now the sum of the "potential energy" {i.e. the objective function E(w)} and the fictitious kinetic energy) of the system; this uses standard finite step size integration techniques. In classical statistical mechanics, the kinetic energy is trivially related to the temperature of the system. In both methods, the system is slowly cooled to zero temperature. This is the so-called annealing schedule. Clearly, the longer the annealing schedule, the more likely is the system to find the absolute minimum, or at least a good, stable minimum. The inclusion of a fictitious temperature allows the system to float over hills in the "energy" landscape. Both schemes will scale as O(N). In the Monte Carlo implementation, forces (i.e. derivatives of the objective function) are not required; in the molecular dynamics implementation they are. The advantage of the latter is that the use of forces provides a more intelligent way of exploring the energy landscape. A typical procedure would be to try different annealing schedules from different starting points and use the best final set of weights after the quenches are completed.

## 5) Genetic Algorithms

Another exploitation of nature's optimization techniques is the genetic algorithm[e,f]. Here the neural network problem of finding extrema resembles natural evolution: a population of possible networks exists, and the rules of natural selection are applied to reproduction, growth, death, etc. The population comprises p different networks, each with its own set of weights and each with its own value for the objective function E(w). This population is

[e] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison Wesley (1989).

[f] L. Davis, *Handbook of Genetic Algorithms*; Van Nostrand Reinhold (1991).

ranked according to a degree of "fitness" according to its objective function. Small values of the objective function are "good." Members of this population are then allowed to mate with one another. Traits, represented by a weight or set of weights within one network, are allowed to exchange with those in another. Networks are chosen at random for the mating process, but the choice is weighted so that the "better" networks mate with one another. Children are then produced and added to the list; the "worst" members of the population are "killed off" by deletion from the list. The population size P is kept fixed. Other genetic "operators" are possible: mutation, direct reproduction with no exchange of traits, and so on. After application of these operators, the new list is reranked and the whole process is reapplied.

Historically, genetic algorithms have been applied to binary strings, whereupon a more direct analogy with genes represents the variables in a given problem. But the same ideas may be equally applied to floating point information and even strings of symbols. The population evolves in such a way that it finds the most important ("best") parts of configuration space. Mutation and cross-over (of trait) operators allow the members of the population to migrate across distant parts of the objective function landscape. Movements across "mountains" are then quite possible. "Intelligence" is injected into the solution since genetic operators ensure that the population evolves toward "good" solutions without deleterious effects from forces and/or "mountains" in the objective landscape.

The problem with genetic algorithms is that, although they are very powerful, they are NOT a "black box." Experimentation is necessary with the frequency at which operators are applied, the types of operators (in principle, they could include gradient information also), the way in which "good" members of the population are selected for mating, how many weights (traits) may be exchanged or mutated, and how new children are inserted into the population. Nevertheless, this method has been applied to neural networks and is found to be superior to most of the foregoing methods for some problems.

It is possible to use a genetic algorithm to find a "good" optimum for the problem, and then use a conjugate gradient algorithm to obtain the true optimum. In fact, this strategy has been demonstrated to work well. Lastly, genetic operators may also be used to design a network: the topology of the network itself may be a variable in the problem. Nodes may be added or subtracted according to genetic "rules" in order to improve the degree of fitness of the population. This is an ongoing area of research; initial indications are that this may be a fruitful avenue to pursue.

## 6) Diffusion Equation Method

The existence of multiple minima represents a challenge in obtaining a global minimum for any of the above methods. Convergence and testing of many minima represent time consuming operations. A method of improving the search for a global minimum using a relatively simple modification of the function under examination has been introduced[g]. By adding increments of the second derivative of the function under investigation, a number of the local minima disappear. Shallow minima disappear before deep ones. Searching for minima in this deformed function is much easier, since there are many fewer minima. Once the global minimum has been located for the modified function, the reverse operation is performed to regenerate the original function while simultaneously following the position of the global minimum to its new location. This procedure appears to be an effective way to determine the location of global minima in complex functions.

## 7) Summary

There are other optimization schemes reported in the literature which may be applied to neural network learning, methods for smoothing out mountains in the "energy" landscape or for constructing simplexes [Footnote b] around an estimated minimum, but the examples given above are representative and illustrative of the inherent difficulty of the problem. Other schemes which borrow from nature might be envisaged, such as designing the problem to resemble the manner in which insects find food. In this case, "swarms" of networks could migrate about in the energy landscape. Once "food" is located (i.e. a low value for the objective function), then a swarm would convene in this region and continue the search. The "swarm" operators here look quite similar to genetic operators; similar success is found in locating the minimum of the objective function.

---

g L. Piela, J. Kostrowicki, and H. A. Scherga, *The Multiple-Minima Problem in the Conformational Analysis of Molecules. Deformation of the Potential Energy Hypersurface by the Diffusion Equation*, J. Phys. Chem. 93, 3339-3346 (1989).

## APPENDIX III. NEURAL NETWORK OPTIMIZATION METHODS

Network learning constitutes a difficult nonlinear problem. However, the methods described above (d and e in particular) may be introduced with significant success, especially when coupled with a final conjugate gradient minimization. Indeed, these methods have been used to solve the "travelling salesman" problem. An important conclusion is that, although an absolute minimum is not guaranteed, there exist "good" solutions in the "energy" landscape which yield acceptable and robust solutions for many problems. Indeed, there is a growing body of work in the theoretical computer science literature which supports the notion that the degree of difficulty associated with "NP-complete" problems is associated with the pathological parts of the configuration space of the problem, which may just be a vanishingly small part of the overall configuration space. In other words, for many problems, the "not quite so good" solutions may not be that bad and, to the contrary, may have significant utility.

# INDEX